Bank customer churn analysis

In [ ]:
```
In the rapidly evolving banking sector, customer retention has become a crit
that influence customer decisions to stay with or leave their banking servic
various attributes of bank customers to identify key predictors of customer
insights that could help devise strategies to enhance customer retention and
```

In [1]:
```python
#Importing necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [196…
```python
df = pd.read_csv('https://drive.google.com/uc?export=download&id=1xh7D0NDmxc
df.head()
```

Out[196…

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
0	1	15634602	Hargrave	619	France	Female	42
1	2	15647311	Hill	608	Spain	Female	41
2	3	15619304	Onio	502	France	Female	42
3	4	15701354	Boni	699	France	Female	39
4	5	15737888	Mitchell	850	Spain	Female	43

In [11]:
```python
df.shape
```

Out[11]: (10000, 18)

In [ ]:
```python
# RowNumber, surname and CustomerId is irrelevant, lets delete it
```

In [197…
```python
df.drop(['RowNumber','CustomerId','Surname'], axis = 1, inplace = True)
df.head()
```

Out[197…

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
0	619	France	Female	42	2	0.00	1
1	608	Spain	Female	41	1	83807.86	1
2	502	France	Female	42	8	159660.80	3
3	699	France	Female	39	1	0.00	2
4	850	Spain	Female	43	2	125510.82	1

In [13]:
```
df.columns
```

Out[13]:
```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
       'Exited', 'Complain', 'Satisfaction Score', 'Card Type',
       'Point Earned'],
      dtype='object')
```

In [14]:
```
df.dtypes
```

Out[14]:
```
CreditScore             int64
Geography              object
Gender                 object
Age                     int64
Tenure                  int64
Balance               float64
NumOfProducts           int64
HasCrCard               int64
IsActiveMember          int64
EstimatedSalary       float64
Exited                  int64
Complain                int64
Satisfaction Score      int64
Card Type              object
Point Earned            int64
dtype: object
```

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CreditScore         10000 non-null  int64  
 1   Geography           10000 non-null  object 
 2   Gender              10000 non-null  object 
 3   Age                 10000 non-null  int64  
 4   Tenure              10000 non-null  int64  
 5   Balance             10000 non-null  float64
 6   NumOfProducts       10000 non-null  int64  
 7   HasCrCard           10000 non-null  int64  
 8   IsActiveMember      10000 non-null  int64  
 9   EstimatedSalary     10000 non-null  float64
 10  Exited              10000 non-null  int64  
 11  Complain            10000 non-null  int64  
 12  Satisfaction Score  10000 non-null  int64  
 13  Card Type           10000 non-null  object 
 14  Point Earned        10000 non-null  int64  
dtypes: float64(2), int64(10), object(3)
memory usage: 1.1+ MB
```

In [16]:

```
df.isna().sum()
```

Out[16]:

```
CreditScore           0
Geography             0
Gender                0
Age                   0
Tenure                0
Balance               0
NumOfProducts         0
HasCrCard             0
IsActiveMember        0
EstimatedSalary       0
Exited                0
Complain              0
Satisfaction Score    0
Card Type             0
Point Earned          0
dtype: int64
```

In [17]:

```
df.duplicated().sum()
```

Out[17]:

```
0
```

In [198…

```
df['HasCrCard'].replace({0 : 'No', 1 : 'Yes'}, inplace = True)
df['IsActiveMember'].replace({0 : 'No', 1 : 'Yes'}, inplace = True)
df['Exited'].replace({0 : 'No', 1 : 'Yes'}, inplace = True)
df['Complain'].replace({0 : 'No', 1 : 'Yes'}, inplace = True)
df.head()
```

Out[198…

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
0	619	France	Female	42	2	0.00	1
1	608	Spain	Female	41	1	83807.86	1
2	502	France	Female	42	8	159660.80	3
3	699	France	Female	39	1	0.00	2
4	850	Spain	Female	43	2	125510.82	1

In [200…

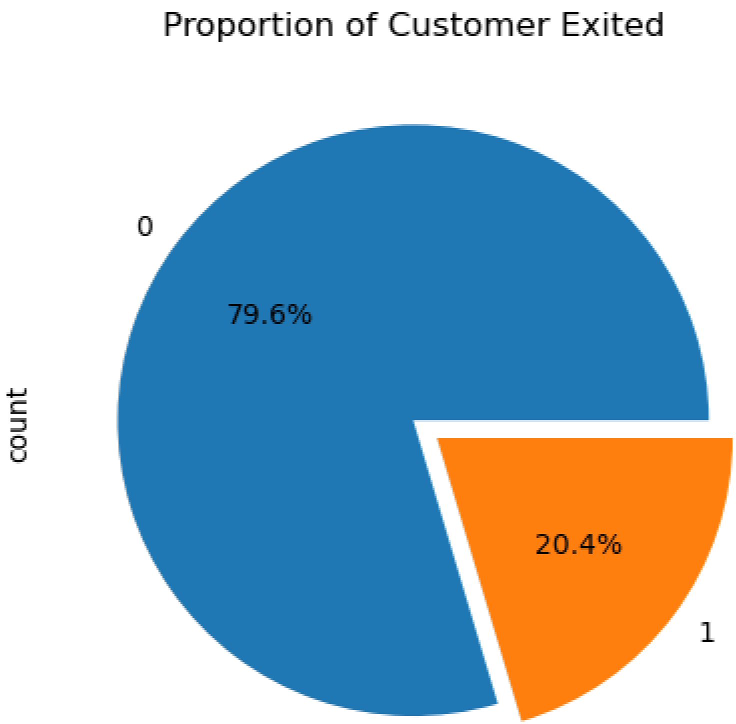
```
columnss = ['Geography', 'Gender', 'NumOfProducts', 'HasCrCard', 'IsActiveMe
for i in columnss:
 print(df.value_counts(i))
 print('\n')
```

```
Geography
France     5014
Germany    2509
Spain      2477
Name: count, dtype: int64


Gender
Male      5457
Female    4543
Name: count, dtype: int64


NumOfProducts
1    5084
2    4590
3     266
4      60
Name: count, dtype: int64


HasCrCard
Yes    7055
No     2945
Name: count, dtype: int64


IsActiveMember
Yes    5151
No     4849
Name: count, dtype: int64


Exited
No     7962
Yes    2038
Name: count, dtype: int64


Complain
No     7956
Yes    2044
Name: count, dtype: int64


Satisfaction Score
3    2042
2    2014
4    2008
5    2004
1    1932
Name: count, dtype: int64


Card Type
DIAMOND     2507
```

```
GOLD        2502
SILVER      2496
PLATINUM    2495
Name: count, dtype: int64
```

In [133…

```
# Check proportion of customer exited
data['Exited'].value_counts().plot.pie(autopct = '%.1f%%', explode =
(0,0.1))
plt.title('Proportion of Customer Exited')
plt.show()
```



20.4 % of the customer have exited the bank

Exploratory Data Analysis (EDA)

Statistical Summary

In [143…

```
#Non Graphical Analysis

columnss = ['Geography', 'Gender', 'NumOfProducts', 'HasCrCard', 'IsActiveMe
for i in columnss:
 print(df.value_counts(i))
 print('\n')
```

Loading [MathJax]/extensions/Safe.js

```
Geography
0    5014
1    2509
2    2477
Name: count, dtype: int64


Gender
1    5457
0    4543
Name: count, dtype: int64


NumOfProducts
1    5084
2    4590
3     266
4      60
Name: count, dtype: int64


HasCrCard
1    7055
0    2945
Name: count, dtype: int64


IsActiveMember
1    5151
0    4849
Name: count, dtype: int64


Exited
0    7962
1    2038
Name: count, dtype: int64


Complain
0    7956
1    2044
Name: count, dtype: int64


Satisfaction Score
3    2042
2    2014
4    2008
5    2004
1    1932
Name: count, dtype: int64


Card Type
0    2507
```

```
1    2502
3    2496
2    2495
Name: count, dtype: int64
```

In [ ]:
```
50% Customers are from France
70% Customer have Credit Card
There seems to be more Male as compared to Females but by small margin
Complain and Exited seems to have some corelation since they have same numbe
Marginally have large number of active members as compared to non-active mem
```

In [ ]:
```
Observations:
Columns NumOfProducts, HasCrCard, IsActiveMember, and Exited were changed to
easier.
Columns RowNumber, CustomerId, CreditScore, Age, Tenure, Balance, and Estima
There are no missing values in the dataset.
This dataset has 14 columns and 1000 rows
```

In [155…
```
#Grouping data for nums and cats
nums = ['CreditScore','Age','Tenure','Balance','EstimatedSalary']
cats = ['Geography','Gender','Exited','HasCrCard','IsActiveMember','NumOfPro
```

In [139…
```
# Automatically detect numerical columns in the DataFrame
nums = df.select_dtypes(include=['number']).columns.tolist()
# Get descriptive statistics for these numerical columns
df[nums].describe()
```

Out[139…

	CreditScore	Geography	Gender	Age	Tenure
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	650.528800	0.746300	0.545700	38.921800	5.012800
std	96.653299	0.827529	0.497932	10.487806	2.892174
min	350.000000	0.000000	0.000000	18.000000	0.000000
25%	584.000000	0.000000	0.000000	32.000000	3.000000
50%	652.000000	0.000000	1.000000	37.000000	5.000000
75%	718.000000	1.000000	1.000000	44.000000	7.000000
max	850.000000	2.000000	1.000000	92.000000	10.000000

Obervation result : CreditScore, Estimatedsalary, and Tenure seem to have a fairly symmetrical distribution of data (mean and median are not much different). Balance is left skewed (means less than the median) and Age is right skewed (means greater than the median) so further observations are needed. If it's skewed then feature transformation will be performed (normalization/standardization or log transformation) on pre-processing data

In [149…

```
#Categorical columns
df[cats].describe()
```

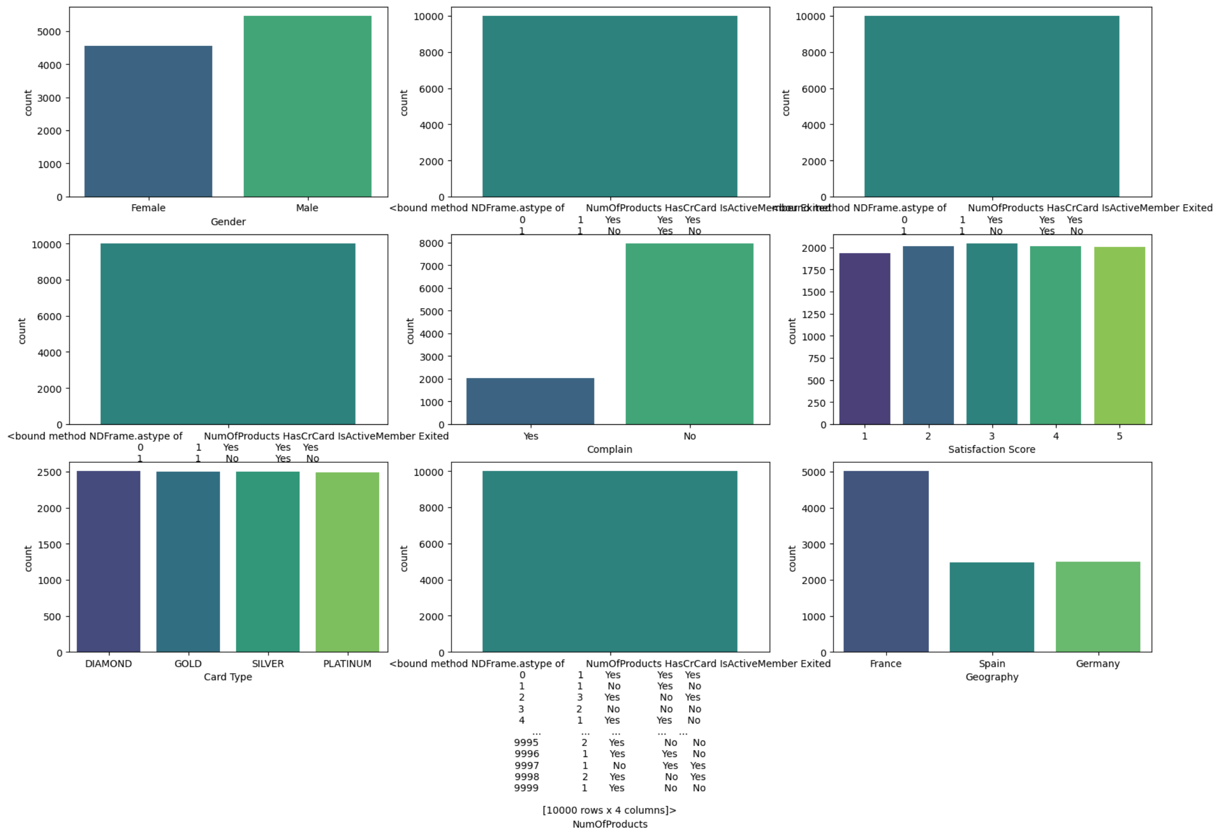
Out[149…

	Geography	Gender
count	10000.000000	10000.000000
mean	0.746300	0.545700
std	0.827529	0.497932
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	1.000000
75%	1.000000	1.000000
max	2.000000	1.000000

Univariate Analysis:

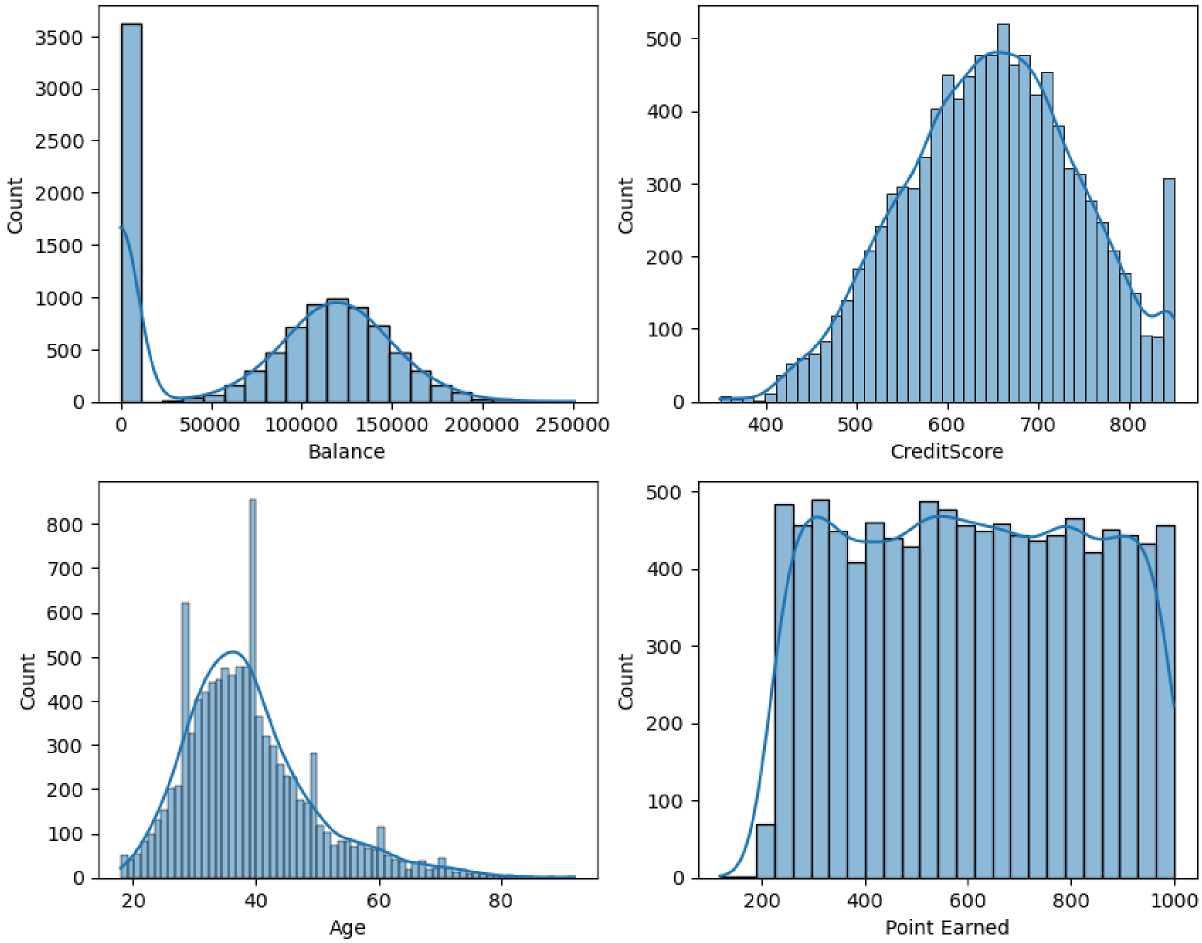
In [202…

```
fig, axs = plt.subplots(nrows = 3, ncols = 3, figsize = (20,12))
sns.countplot(data = df, x = 'Gender', ax = axs[0,0], palette = 'viridis')
sns.countplot(data = df, x = 'HasCrCard', ax = axs[0,1], palette = 'viridis'
sns.countplot(data = df, x = 'IsActiveMember', ax = axs[0,2], palette = 'vir
sns.countplot(data = df, x = 'Exited', ax = axs[1,0], palette = 'viridis')
sns.countplot(data = df, x = 'Complain', ax = axs[1,1], palette = 'viridis')
sns.countplot(data = df, x = 'Satisfaction Score', ax = axs[1,2], palette =
sns.countplot(data = df, x = 'Card Type', ax = axs[2,0], palette = 'viridis'
sns.countplot(data = df, x = 'NumOfProducts', ax = axs[2,1], palette = 'viri
sns.countplot(data = df, x = 'Geography', ax = axs[2,2], palette = 'viridis'
plt.show()
```



In [151…

```
fig, axs = plt.subplots(nrows = 2, ncols = 2, figsize = (10,8))
sns.histplot(data = df, x = 'Balance', kde = True, ax = axs[0,0])
sns.histplot(data = df, x = 'CreditScore', kde = True, ax = axs[0,1])
sns.histplot(data = df, x = 'Age', kde = True, ax = axs[1,0])
sns.histplot(data = df, x = 'Point Earned', kde = True, ax = axs[1,1])
plt.show()
```



In [ ]:

In [ ]:

Step 1: Descriptive Statistics -

We will be performing descriptive statistics and distribution analysis on key numerical variables in the dataset. We'll calculate the mean, median, and mode for the numerical columns and then visualize their distributions using histograms and box plots.

First, let's calculate the mean, median, and mode for the numerical columns CreditScore, Age, Balance, NumOfProducts, EstimatedSalary, and Point Earned.

Calculate Mean, Median, and Mode

In [19]:

```
from sklearn.preprocessing import LabelEncoder
from scipy import stats
from scipy.stats import norm, chi2_contingency, chisquare, ttest_ind
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

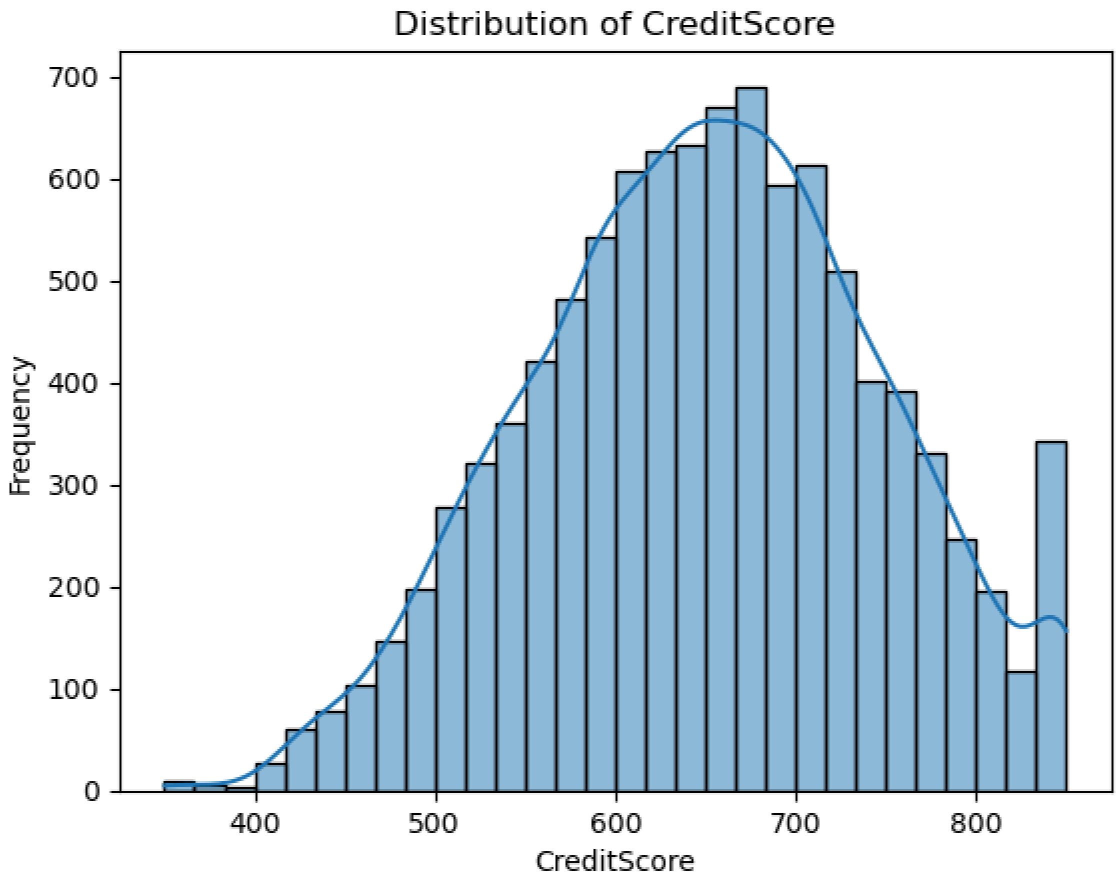
```
# Convert categorical variables to numerical if necessary
categorical_cols = ['Geography', 'Gender']
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# List of numerical columns
numerical_cols = ['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'Estimat

# Calculate and print mean, median, and mode for each numerical column
for col in numerical_cols:
    mean_value = df[col].mean()
    median_value = df[col].median()
    mode_value = df[col].mode()[0]
    print(f"{col} - Mean: {mean_value}, Median: {median_value}, Mode: {mode_
```

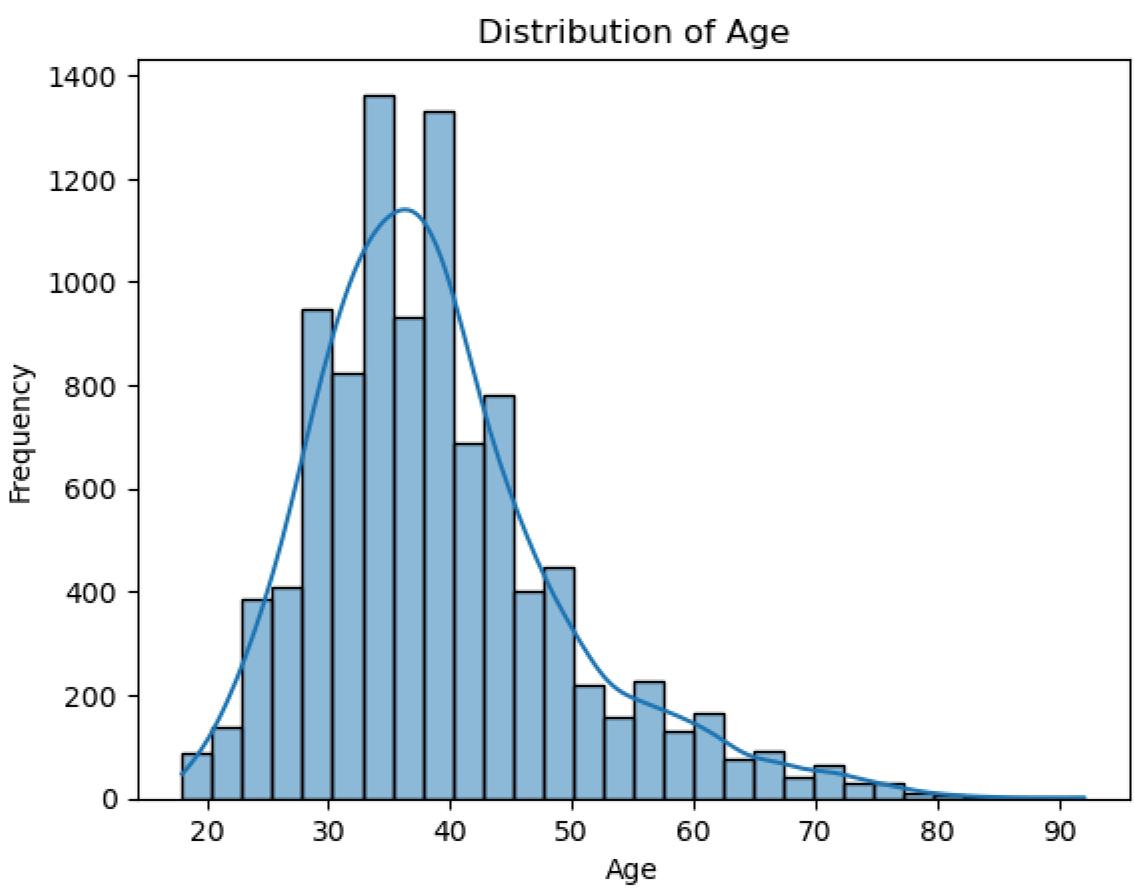
```
CreditScore - Mean: 650.5288, Median: 652.0, Mode: 850
Age - Mean: 38.9218, Median: 37.0, Mode: 37
Balance - Mean: 76485.889288, Median: 97198.54000000001, Mode: 0.0
NumOfProducts - Mean: 1.5302, Median: 1.0, Mode: 1
EstimatedSalary - Mean: 100090.239881, Median: 100193.915, Mode: 24924.92
Point Earned - Mean: 606.5151, Median: 605.0, Mode: 408
```

Step 2: Distribution Analysis

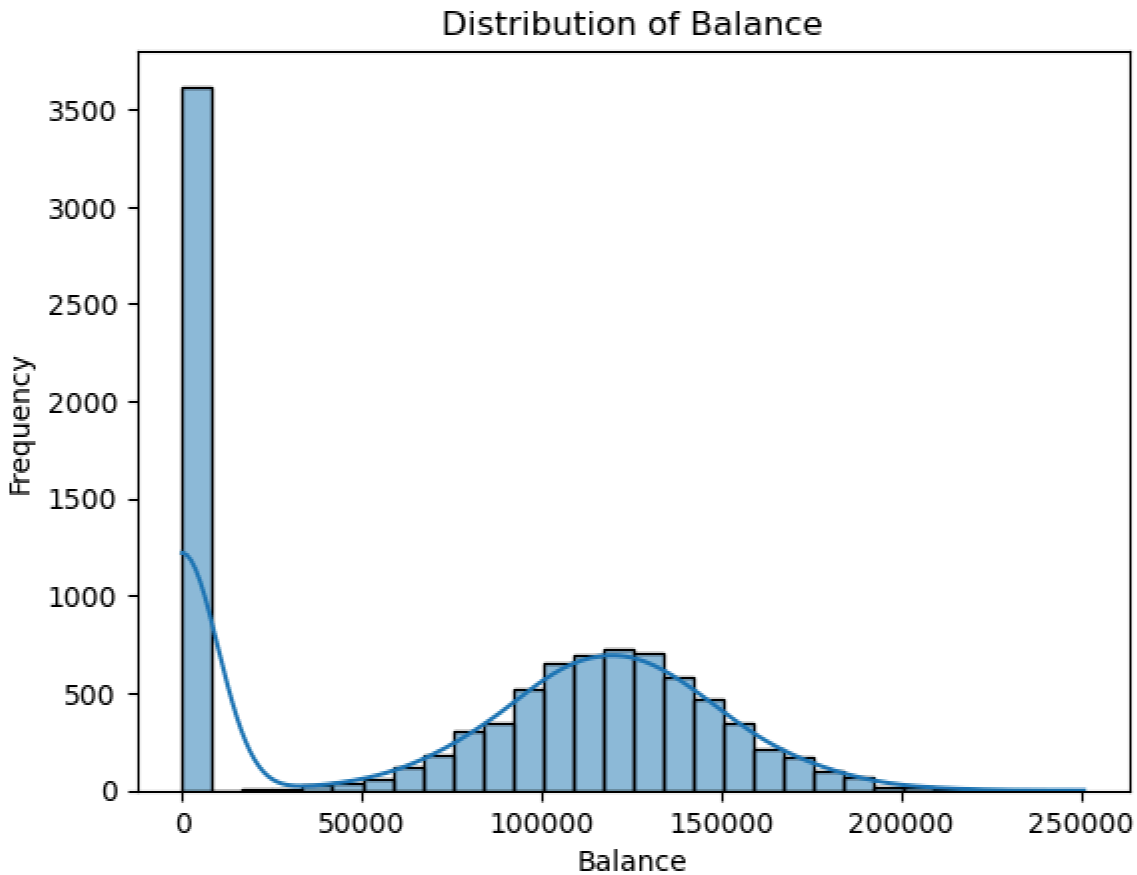
In [23]:

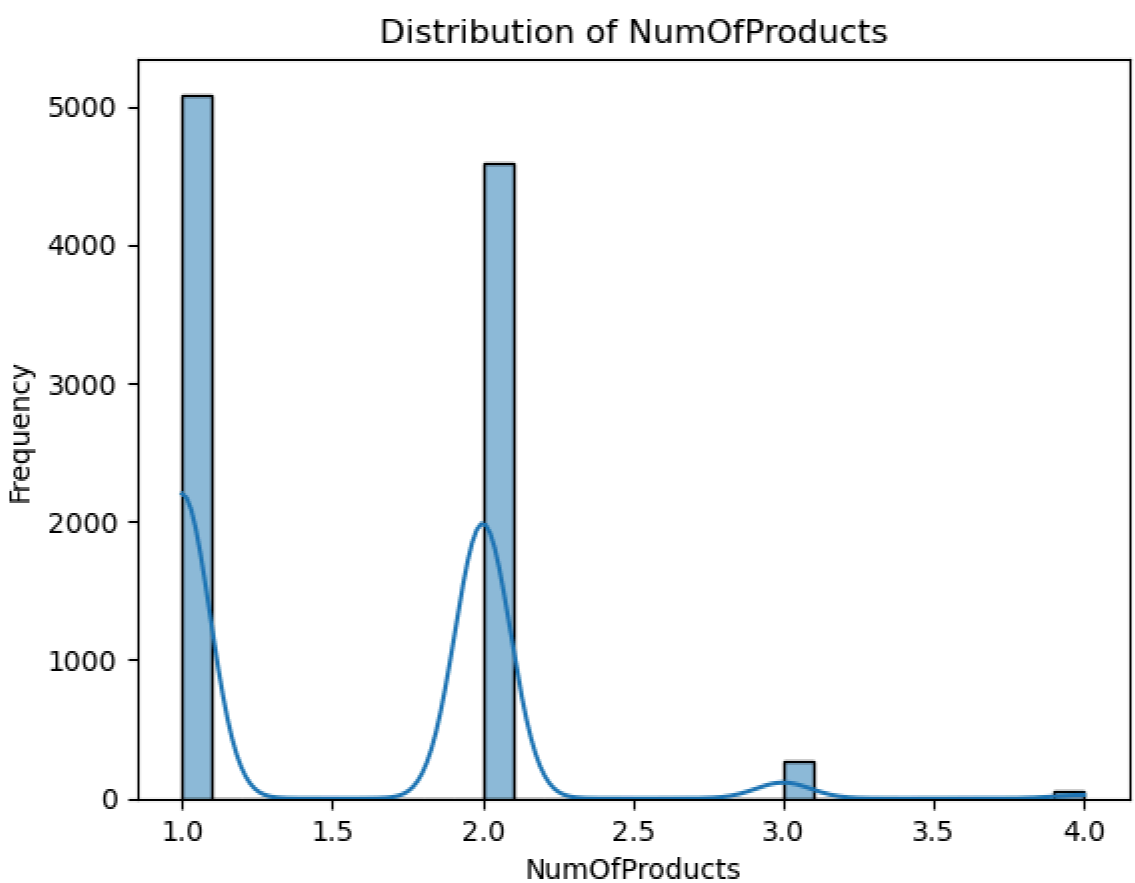
```
#Next, let's analyze the distribution of these numerical variables using his

# Plot histograms for each numerical column
for col in numerical_cols:
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```



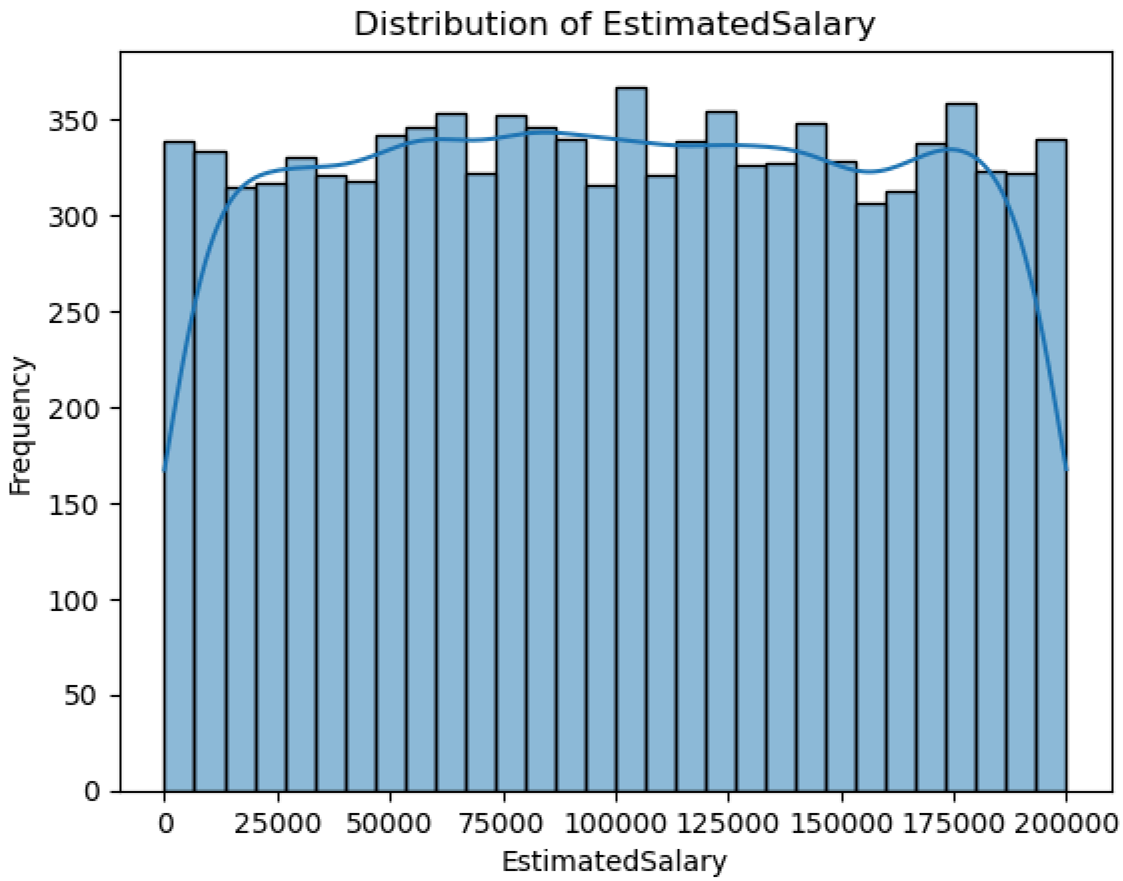


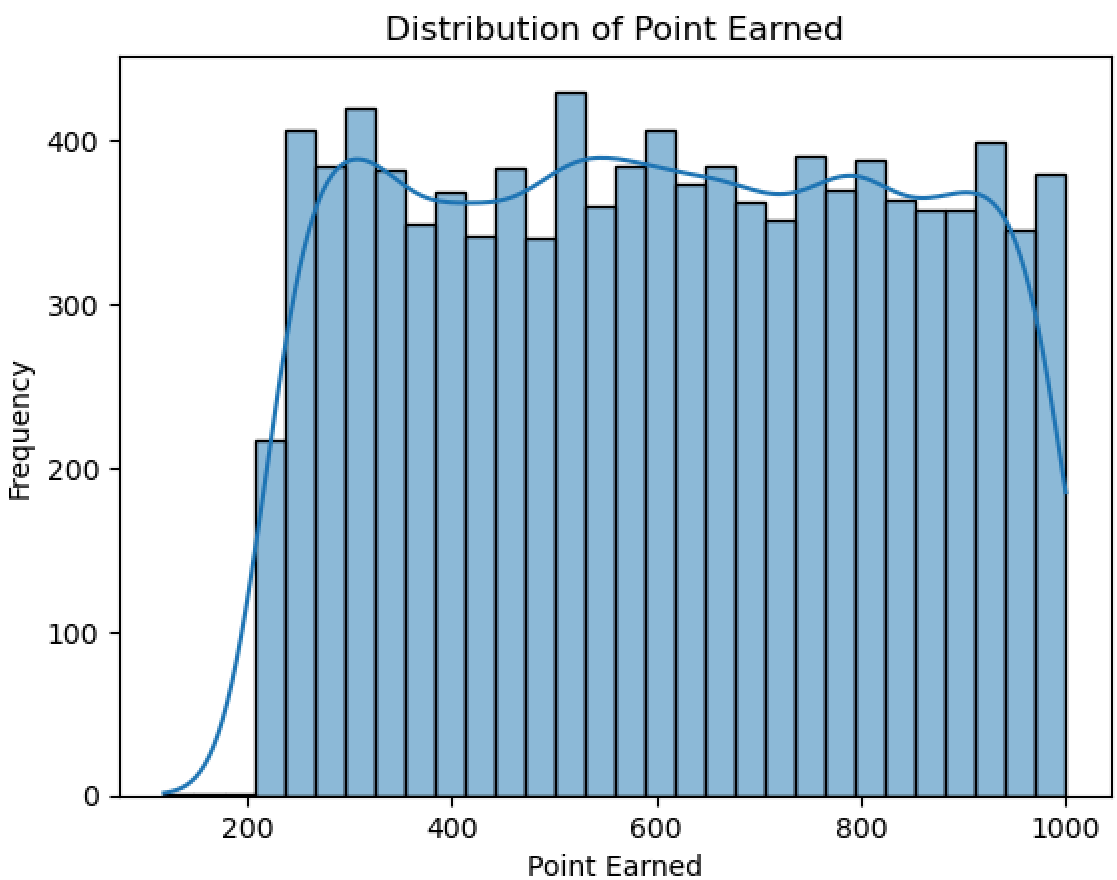
Loading [MathJax]/extensions/Safe.js





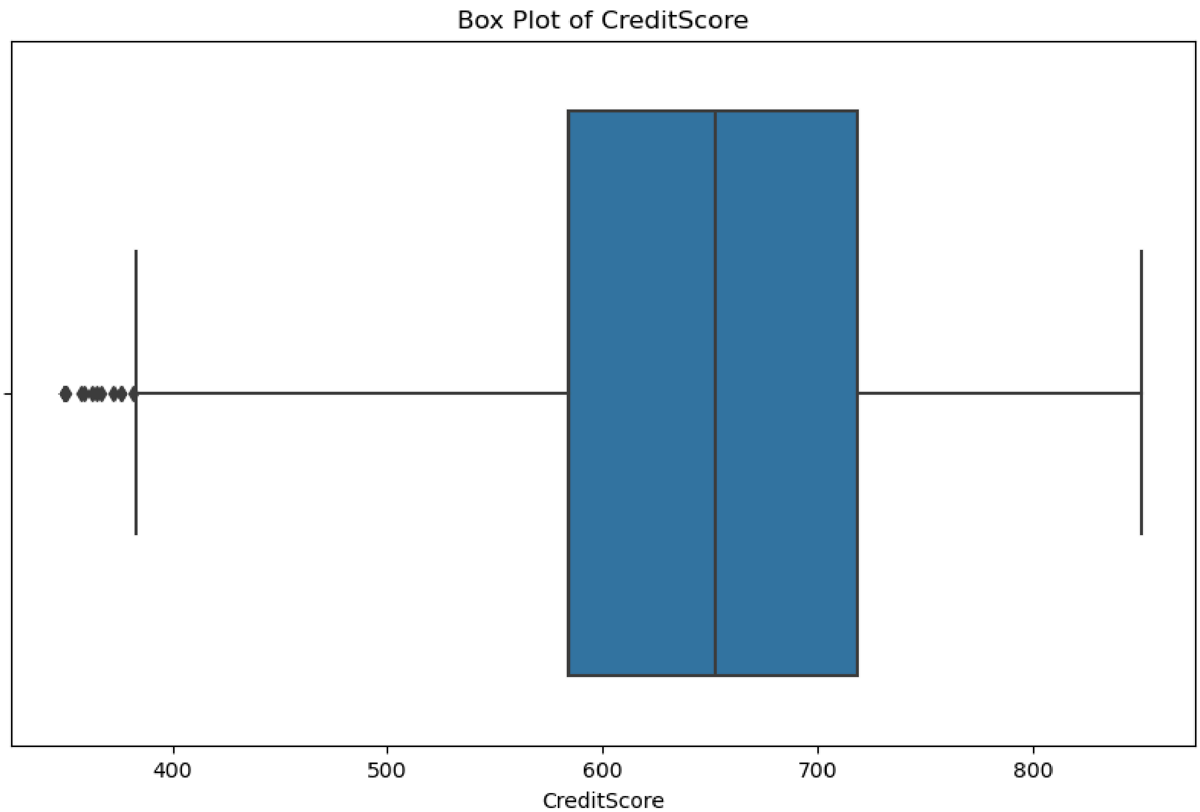
Loading [MathJax]/extensions/Safe.js

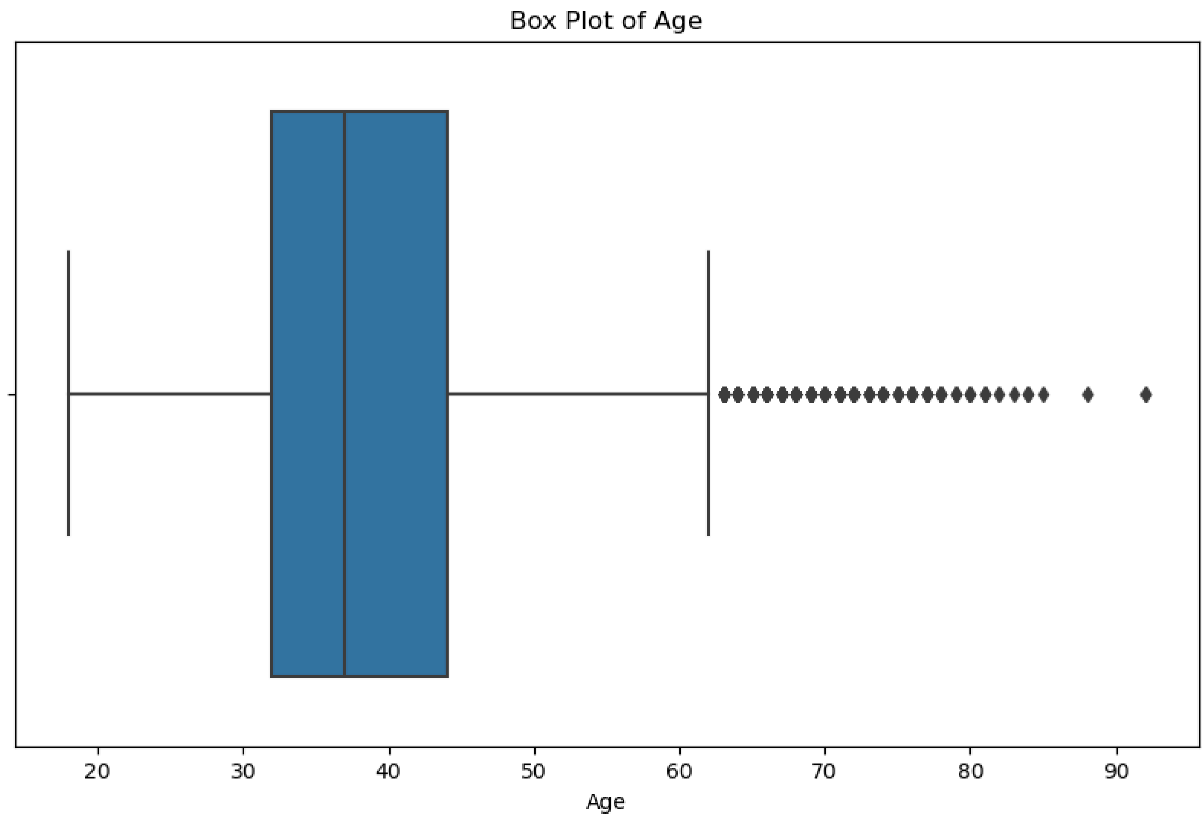


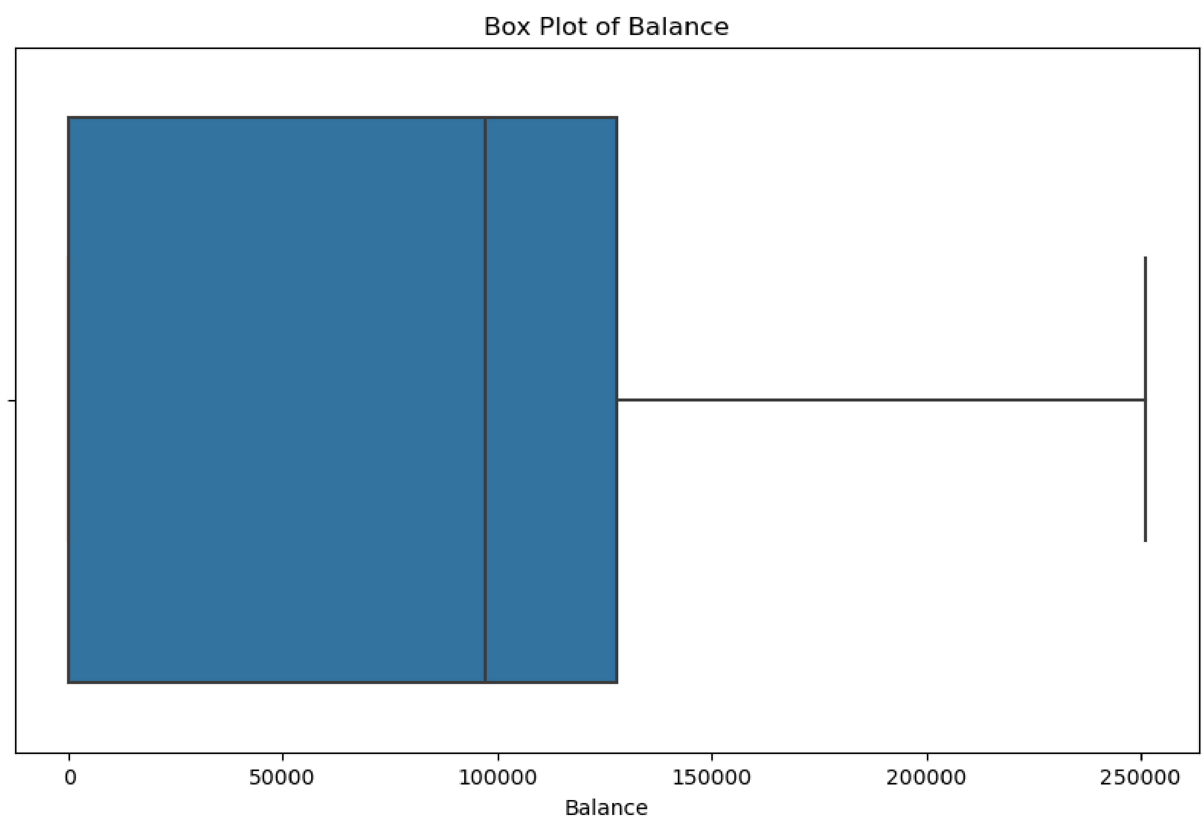


Loading [MathJax]/extensions/Safe.js

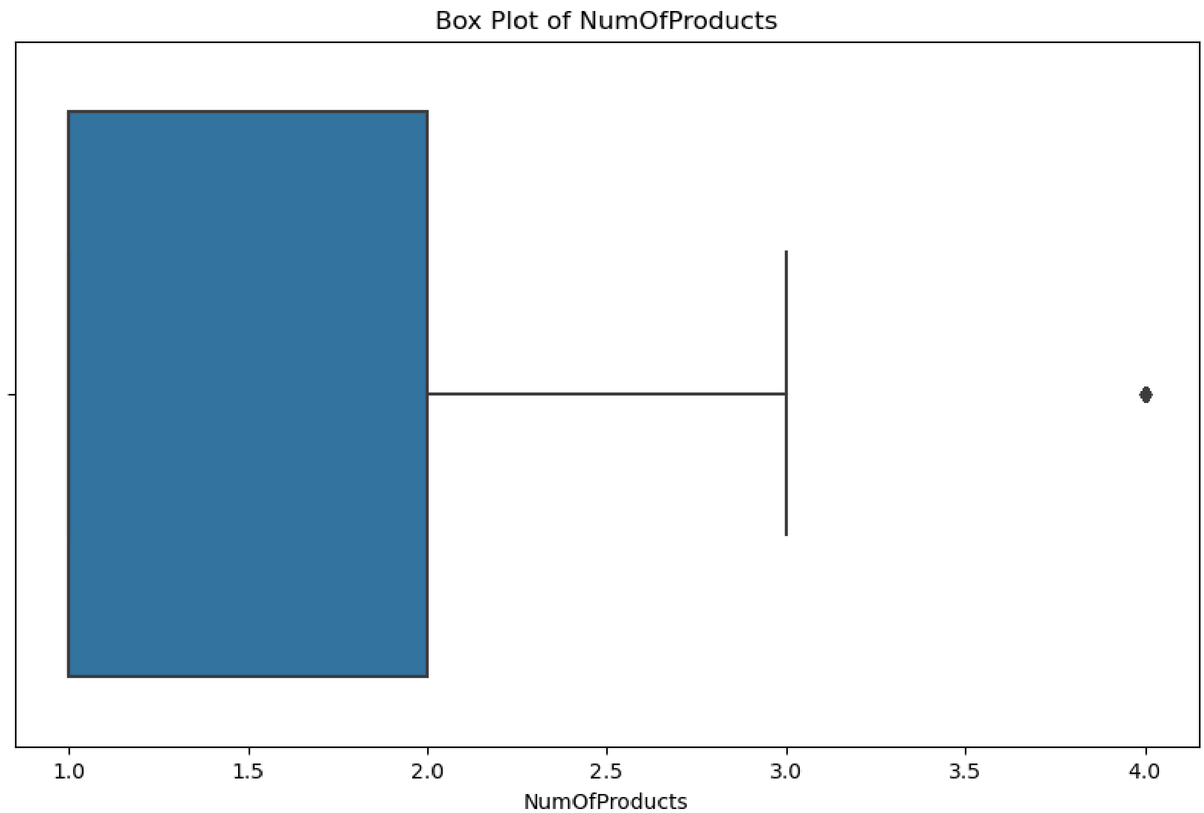
In [41]:
```
# Plot box plots for each numerical column
for col in numerical_cols:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=df[col])
    plt.title(f'Box Plot of {col}')
    plt.xlabel(col)
    plt.show()
```

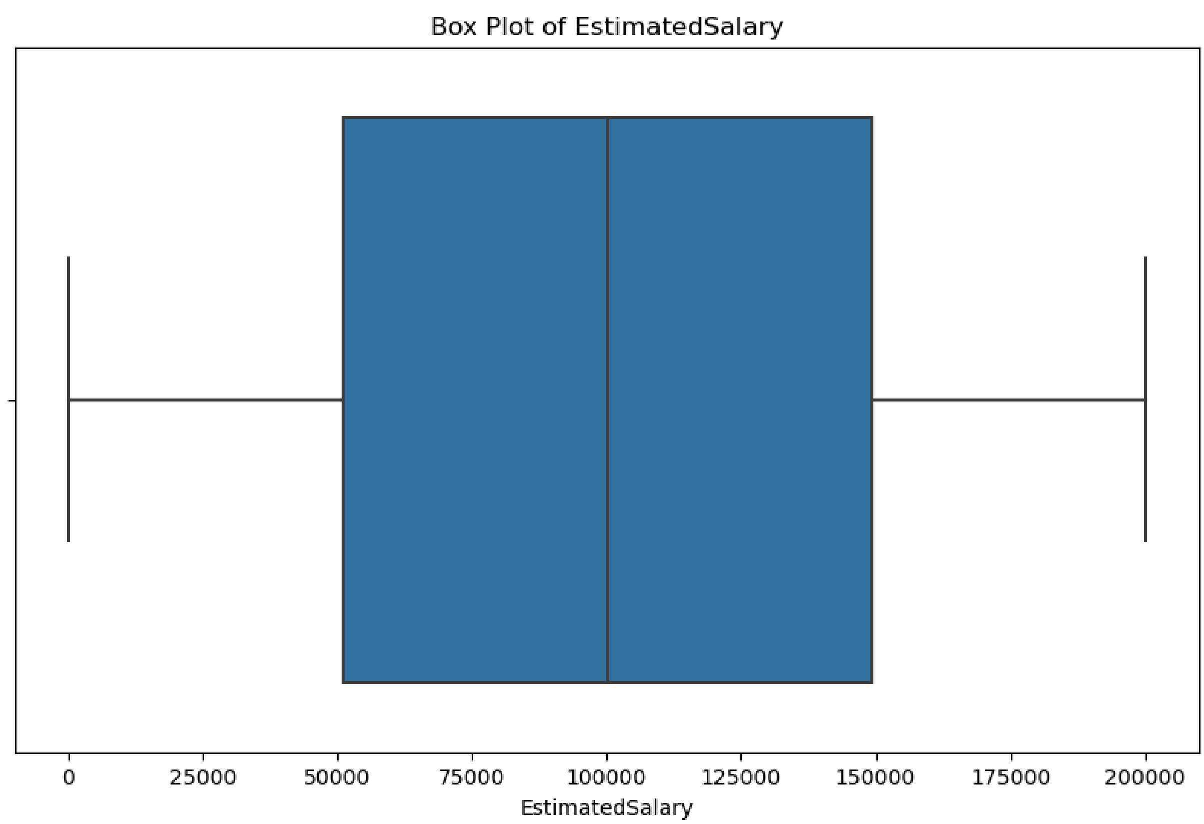


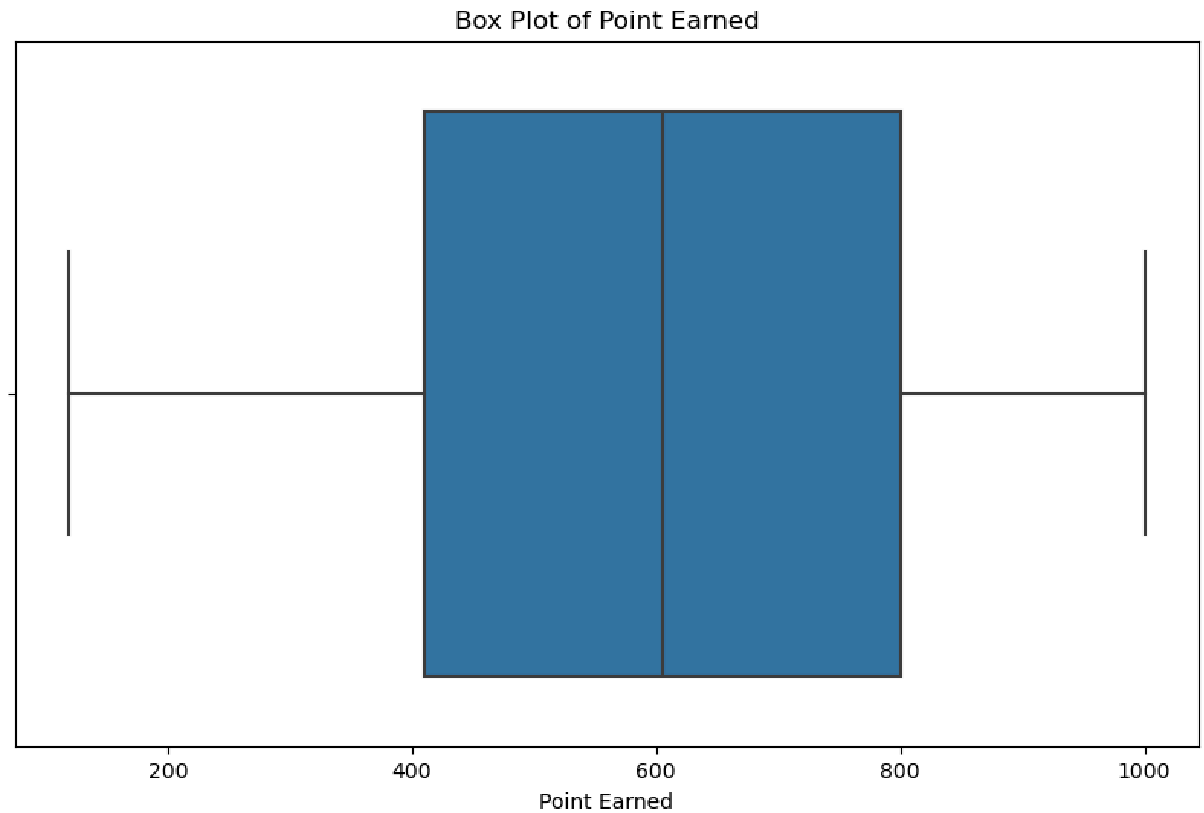




Loading [MathJax]/extensions/Safe.js


Box Plot of NumOfProducts
1.0
1.5
2.0
2.5
3.0
3.5
4.0
NumOfProducts


Box Plot of EstimatedSalary
0
25000
50000
75000
100000
125000
150000
175000
200000
EstimatedSalary



Interpretation Mean, Median, and Mode: These statistics provide insights into the central tendency of the data for each numerical feature.

Mean: The average value of the column. Median: The middle value when the data is sorted. Mode: The most frequently occurring value in the column. Histograms: These visualizations show the distribution of the data, indicating the frequency of different ranges of values. The presence of a KDE (Kernel Density Estimate) line helps to understand the data's probability density function.

CreditScore: The average credit score is around 650, with a standard deviation of about 96. Age: The median age of customers is 39 years, with the majority of customers aged between 29 and 49. Balance: There is a wide range in account balances, with a significant portion of customers having a balance of 0. NumOfProducts: Most customers have 1 or 2 products. EstimatedSalary: The salaries range widely, with an average around 101,322. PointsEarned: The average points earned is around 550, with most customers earning between 463 and 637 points.

Box Plots: These plots provide a summary of the data's distribution, highlighting the median, quartiles, and potential outliers.

In [25]:
```
numerical_columns = ['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'Esti
descriptive_stats = df[numerical_columns].describe()
descriptive_stats
```

Out[25]:

	CreditScore	Age	Balance	NumOfProducts	Estimated
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.0
mean	650.528800	38.921800	76485.889288	1.530200	100090.2
std	96.653299	10.487806	62397.405202	0.581654	57510.4
min	350.000000	18.000000	0.000000	1.000000	11.5
25%	584.000000	32.000000	0.000000	1.000000	51002.1
50%	652.000000	37.000000	97198.540000	1.000000	100193.9
75%	718.000000	44.000000	127644.240000	2.000000	149388.2
max	850.000000	92.000000	250898.090000	4.000000	199992.4

In [157…

```
#Kolmogorov-Smirnov statistical test to see the size of the normality of the
# Interpretation
# If the P-value of the KS Test is larger than 0.05, we assume a normal dist
# If the P-value of the KS Test is smaller than 0.05, we do not assume a nor
from scipy.stats import kstest

for i in df[nums]:
 print(kstest(df[i], 'norm'))
 ks_statistic, ks_pvalue = kstest(df[i], 'norm')
 if ks_pvalue > 0.05:
     print(f'P-value {i}: {ks_pvalue}. So, we assume a normal distribution')
 else:
     print(f'P-value {i}: {ks_pvalue}. So, we do not assume a normal distrib
```

```
KstestResult(statistic=1.0, pvalue=0.0, statistic_location=350, statistic_si
gn=-1)
P-value CreditScore: 0.0. So, we do not assume a normal distribution
KstestResult(statistic=1.0, pvalue=0.0, statistic_location=18, statistic_sig
n=-1)
P-value Age: 0.0. So, we do not assume a normal distribution
KstestResult(statistic=0.8324498680518208, pvalue=0.0, statistic_location=2,
statistic_sign=-1)
P-value Tenure: 0.0. So, we do not assume a normal distribution
KstestResult(statistic=0.6383, pvalue=0.0, statistic_location=3768.69, stati
stic_sign=-1)
P-value Balance: 0.0. So, we do not assume a normal distribution
KstestResult(statistic=1.0, pvalue=0.0, statistic_location=11.58, statistic_
sign=-1)
P-value EstimatedSalary: 0.0. So, we do not assume a normal distribution
```

In [ ]:

```
Based on the Kolmogorov-Smirnov test, all numeric columns don't have normal
be carried out using Box-Cox on pre-processing data.
```

In [ ]:

```
#COUNTPLOT
```

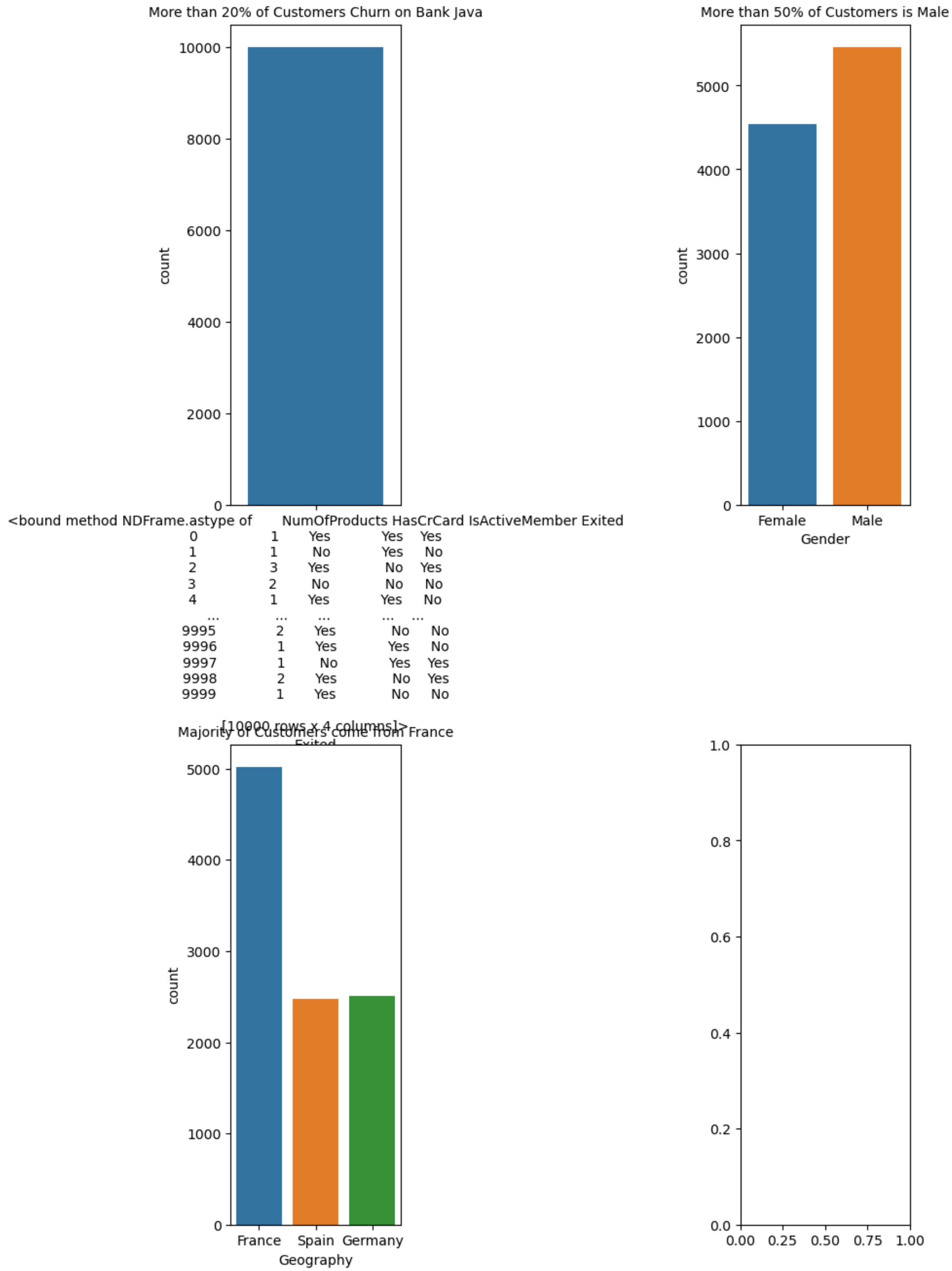
In [219…

```
#Categorical columns data distribution
fig, axes = plt.subplots(2, 2, figsize=(9,16))
sns.countplot(x='Exited', data = df, ax=axes[0][0])
sns.countplot(x='Gender', data = df, ax=axes[0][1])
```

```python
sns.countplot(x='Geography',data = df, ax=axes[1][0])




plt.subplots_adjust(hspace = 0.5, wspace= 2.0)
axes[0][0].set_title('More than 20% of Customers Churn on Bank Java', fontsi
axes[0][1].set_title('More than 50% of Customers is Male', fontsize = 10);
axes[1][0].set_title('Majority of Customers come from France', fontsize = 10
```

Out[219…

```
Text(0.5, 1.0, 'Majority of Customers come from France')
```



In [ ]:

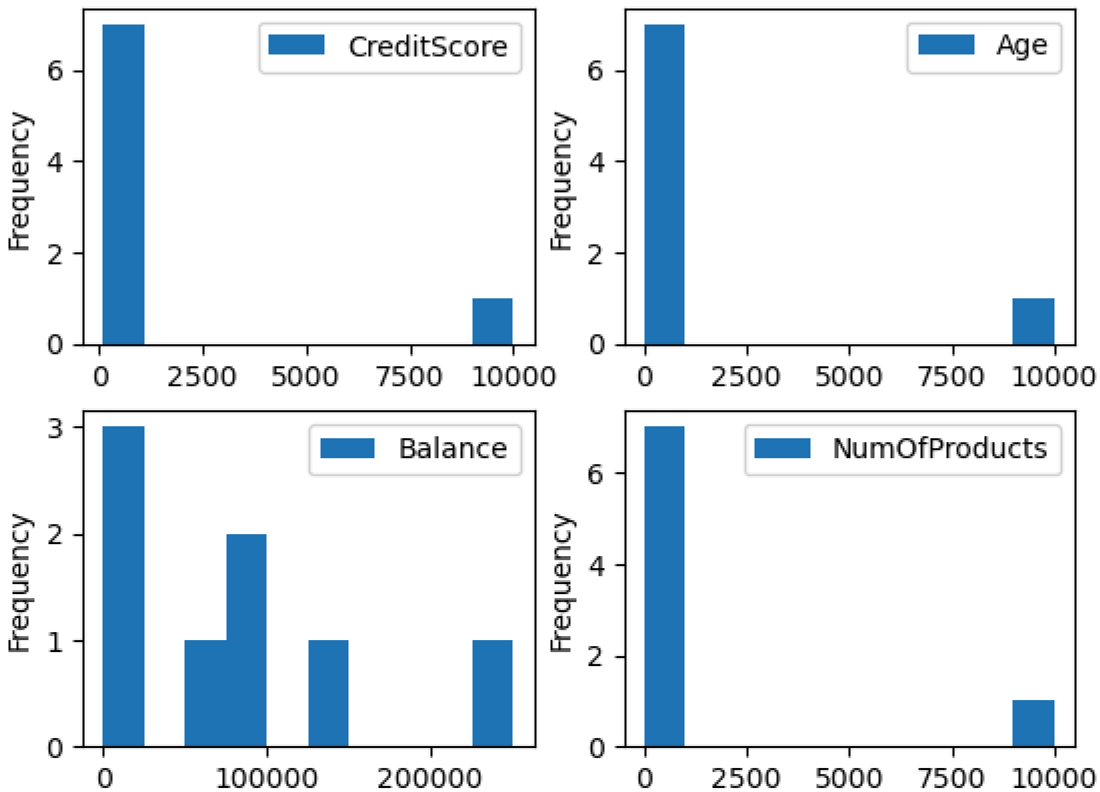
In [ ]: From all customers on Bank Java, there are more than 2000 (>20%) churn custo
compared to customers based on Geography. There is no significant difference
ownership, it can be seen that most customers have credit cards. More than 5
or 2 bank products.

Loading [MathJax]/extensions/Safe.js

In [ ]:
```
#frequency plot of all the numerical colums
```

In [28]:
```
import matplotlib.pyplot as plt

# Create a figure with 2 rows and 2 columns
fig, axes = plt.subplots(2, 2)

# Plot each column in a separate subplot
descriptive_stats.plot(kind="hist", y="CreditScore", ax=axes[0, 0])
descriptive_stats.plot(kind="hist", y="Age", ax=axes[0, 1])
descriptive_stats.plot(kind="hist", y="Balance", ax=axes[1, 0])
descriptive_stats.plot(kind="hist", y='NumOfProducts', ax=axes[1,1])
```

Out[28]: <Axes: ylabel='Frequency'>

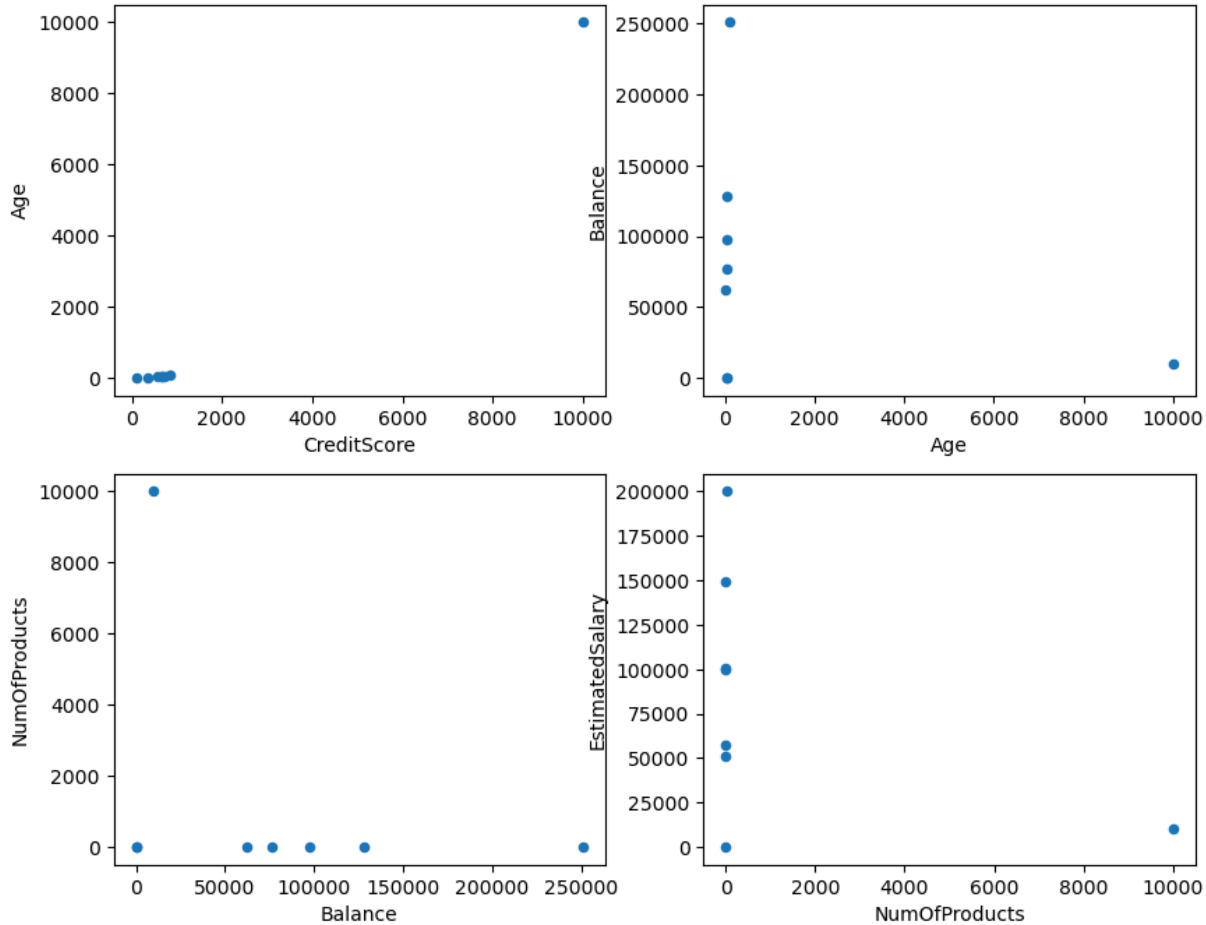


In [ ]:
```
#Lets plot a scatter plot to understand distribution between numerical colum
```

In [36]:
```
# @title CreditScore vs Age

from matplotlib import pyplot as plt
fig, axes = plt.subplots(2, 2,figsize=(10, 8))
descriptive_stats.plot(kind='scatter', x='CreditScore', y='Age',ax=axes[0, 0
descriptive_stats.plot(kind='scatter', x='Age', y='Balance', ax=axes[0, 1])
descriptive_stats.plot(kind='scatter', x='Balance', y='NumOfProducts',ax=axe
descriptive_stats.plot(kind='scatter', x='NumOfProducts', y='EstimatedSalary
```

Out[36]: <Axes: xlabel='NumOfProducts', ylabel='EstimatedSalary'>

Loading [MathJax]/extensions/Safe.js



In [ ]:

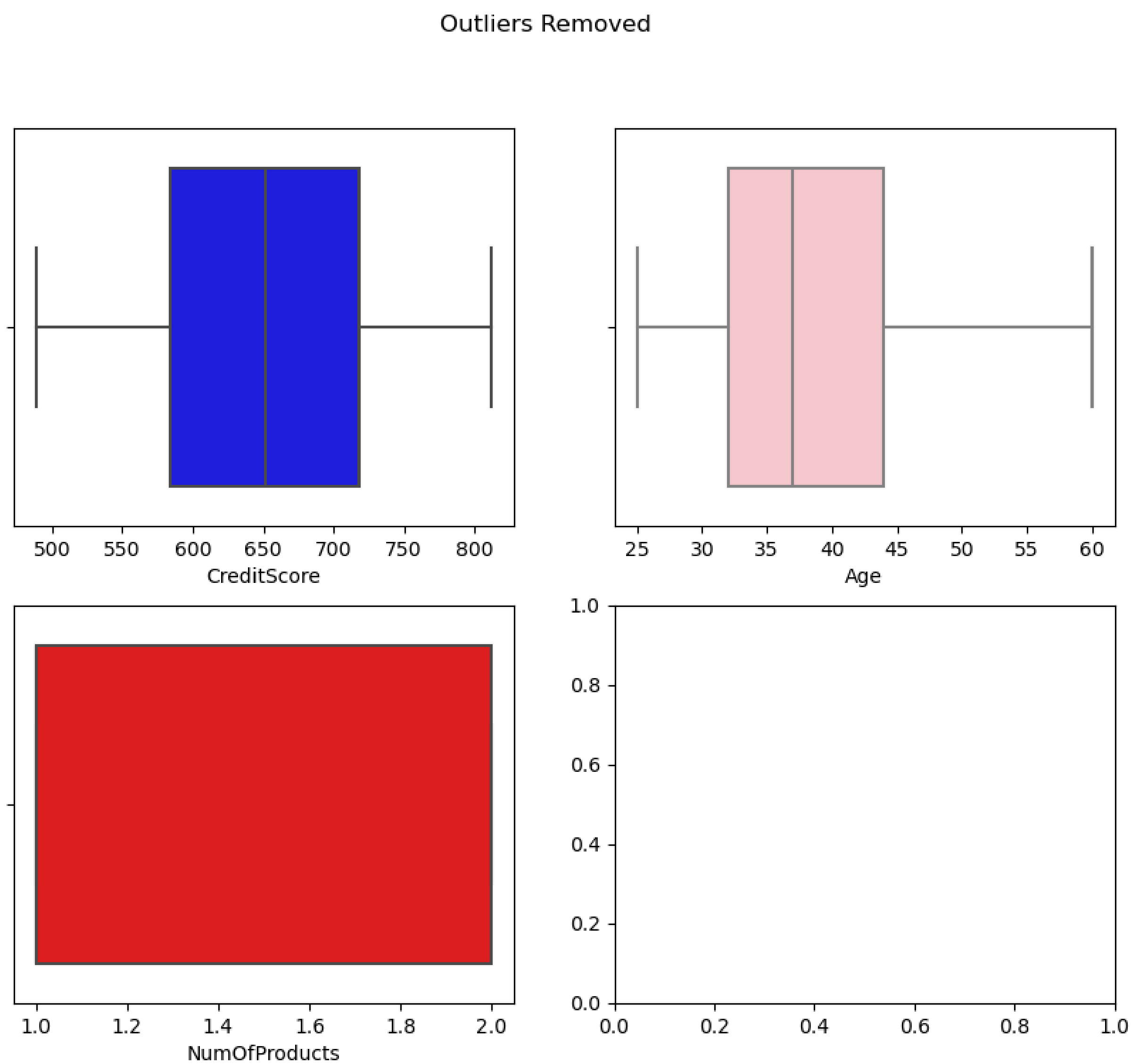
In [ ]:
```
For Numerical variables we can we use Histogram or Boxplot but I have used E
we understand Outliers are present in CreditScore, Age, NumOfProducts.
```

In [ ]:

For Numerical variables we can we use Histogram or Boxplot but I have used Boxplot here to understand the presence of Outliers. By the boxplot, we understand Outliers are present in CreditScore, Age, NumOfProducts.

In [38]:
```
remove_CreditScore = np.clip(df['CreditScore'], np.percentile(df['CreditScor
remove_Age = np.clip(df['Age'], np.percentile(df['Age'],5), np.percentile(df
remove_NumOfProducts = np.clip(df['NumOfProducts'], np.percentile(df['NumOfP
```

In [40]:
```
fig,ax = plt.subplots(2,2, figsize=(10,8))
sns.boxplot(data=df, x=remove_CreditScore,color='blue',ax=ax[0,0])
sns.boxplot(data=df, x=remove_Age, color='pink', ax=ax[0,1])
sns.boxplot(data=df, x=remove_NumOfProducts, color='red', ax=ax[1,0])
plt.suptitle('Outliers Removed')
plt.show()
```



So we Removed Outliers with Clip function in CreditScore, Age, NumOfProducts.

Let's proceed with the more exploratory data analysis (EDA) to understand the relationships and patterns in the data. We'll focus on two main tasks:

Correlation Analysis: Explore the correlation between numerical features and the Exited variable to identify potential predictors of churn.

Customer Profile Analysis: Segment customers based on key demographics (Age, Geography, Gender) to identify which groups are more likely to churn.

In [46]:

```
# Convert categorical variables to numerical if necessary
categorical_cols = ['Geography', 'Gender','Card Type']
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

```python
# Display the first few rows of the DataFrame
print(df.head())
```

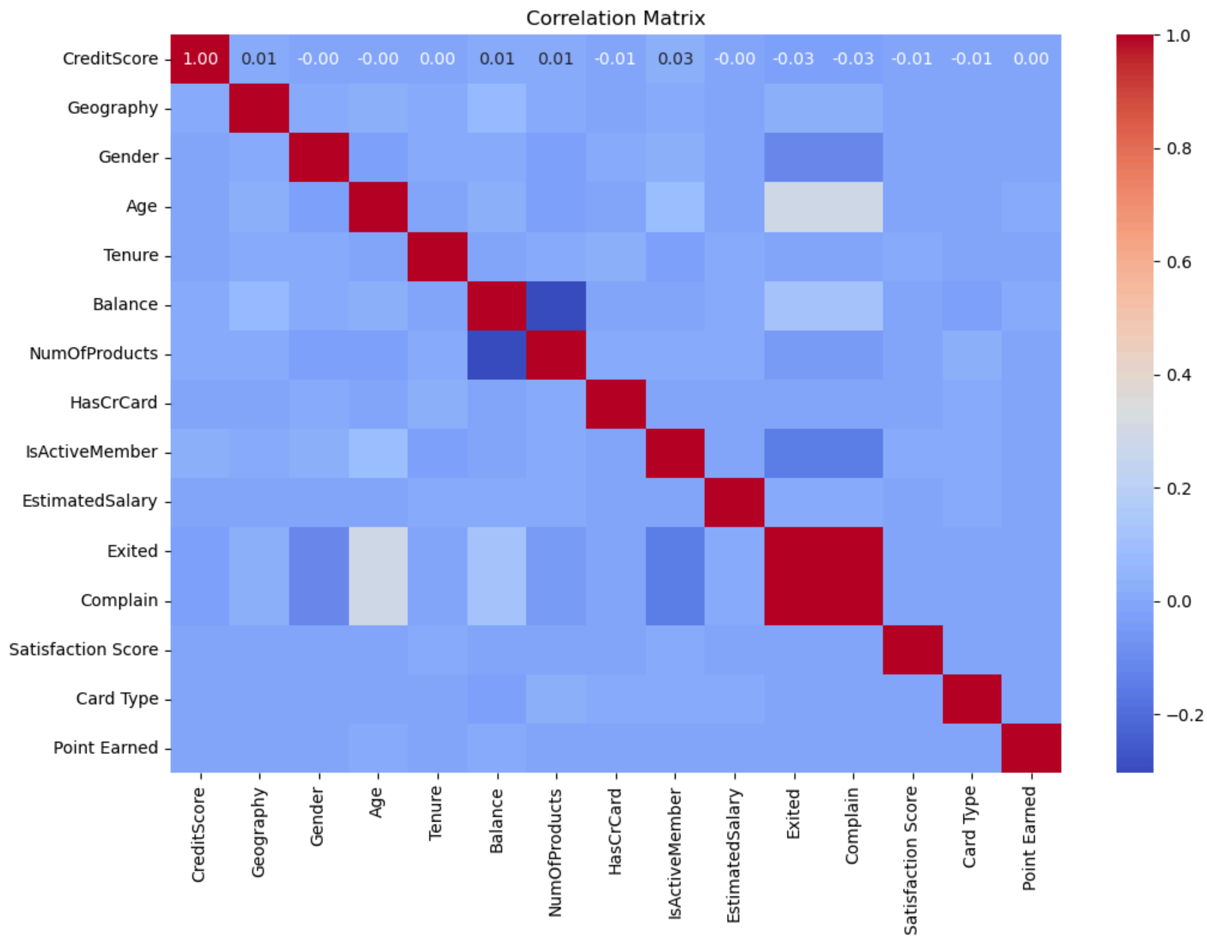
```
   CreditScore  Geography  Gender  Age  Tenure    Balance  NumOfProducts  \
0          619          0       0   42       2       0.00              1   
1          608          2       0   41       1   83807.86              1   
2          502          0       0   42       8  159660.80              3   
3          699          0       0   39       1       0.00              2   
4          850          2       0   43       2  125510.82              1   

   HasCrCard  IsActiveMember  EstimatedSalary  Exited  Complain  \
0          1               1        101348.88       1         1   
1          0               1        112542.58       0         1   
2          1               0        113931.57       1         1   
3          0               0         93826.63       0         0   
4          1               1         79084.10       0         0   

   Satisfaction Score  Card Type  Point Earned  
0                   2          0           464  
1                   3          0           456  
2                   3          0           377  
3                   5          1           350  
4                   5          1           425  
```

Step 2: Correlation Analysis We'll compute the Pearson correlation coefficient between the numerical features and the Exited variable to identify potential predictors of churn.

In [47]:

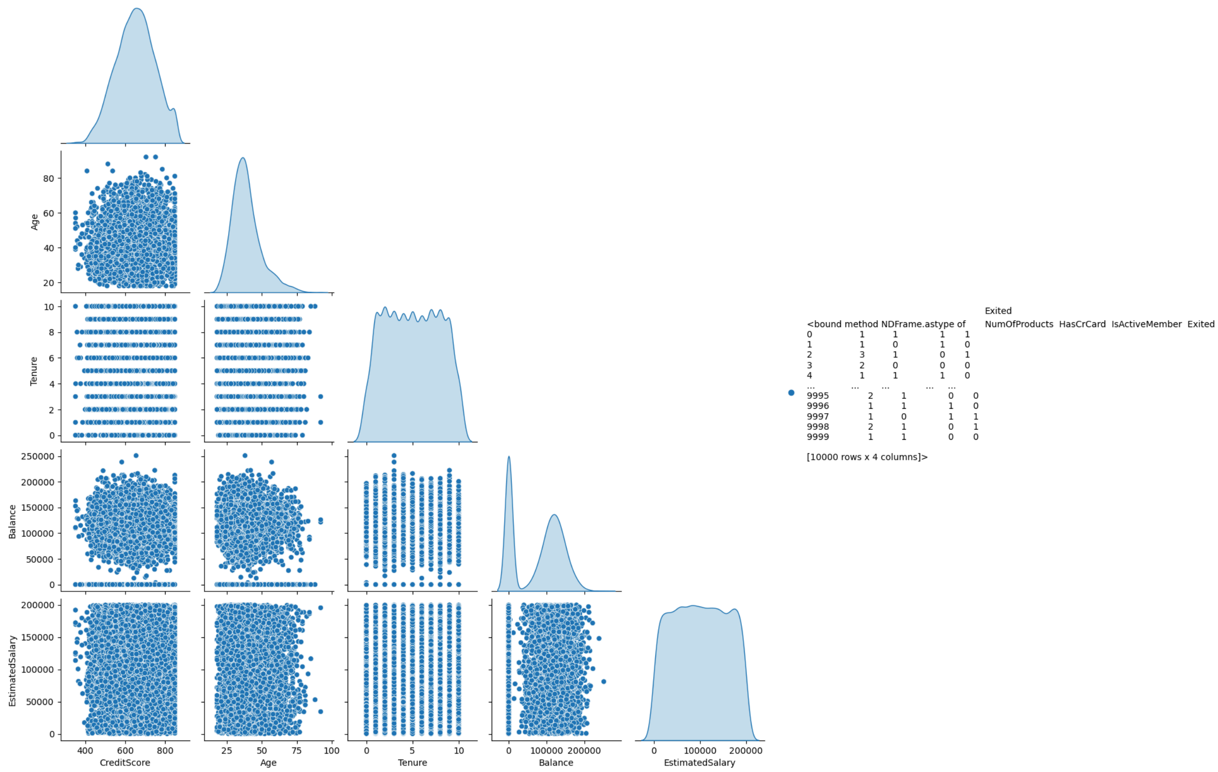
```python
# List of numerical columns
numerical_cols = ['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'Estimat

# Compute Pearson correlation coefficients with the target variable 'Exited

correlation_results = {}

for col in numerical_cols:
    correlation_matrix = np.corrcoef(df[col], df['Exited'])
    correlation = correlation_matrix[0, 1]  # Extract the correlation coeffi
    correlation_results[col] = correlation

# Display the results
for col, correlation in correlation_results.items():
    print(f'Pearson correlation coefficient between {col} and Exited: {corre

# Plot correlation matrix including the 'Exited' variable
corr_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

```
Pearson correlation coefficient between CreditScore and Exited: -0.03
Pearson correlation coefficient between Age and Exited: 0.29
Pearson correlation coefficient between Balance and Exited: 0.12
Pearson correlation coefficient between NumOfProducts and Exited: -0.05
Pearson correlation coefficient between EstimatedSalary and Exited: 0.01
Pearson correlation coefficient between Point Earned and Exited: -0.00
```



In [167…
```
feature = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasC
plt.figure(figsize=(40,10))
sns.pairplot(df[feature], hue='Exited', diag_kind='kde', corner=True)
```

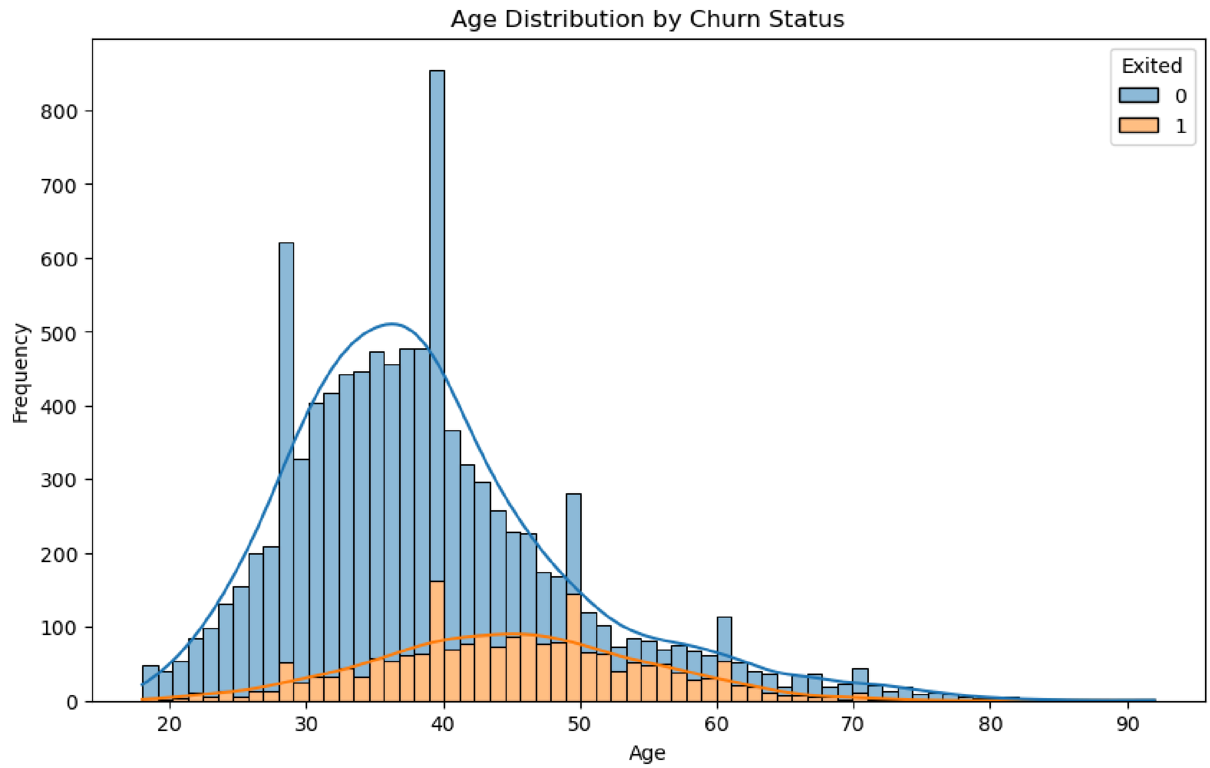
Out[167…
```
<seaborn.axisgrid.PairGrid at 0x200d61d2e50>
<Figure size 4000x1000 with 0 Axes>
```



In [ ]:
```
Based on the results of the Pair Plot, there are several observations:
1. The more separate the Exited and Not Exited values in each column, the be
CreditScore, and Age columns
2. Higher EstimatedSalary and NumOfProduct, higher the probability of custom
3. Higher Balance with NumOfProduct, higher the probability of customer chur
4. Higher Tenure with NumOfProduct, higher the probability of customer churn
```

Step 3: Customer Profile Analysis We'll segment customers based on key demographics (Age, Geography, Gender) to identify which groups are more likely to churn

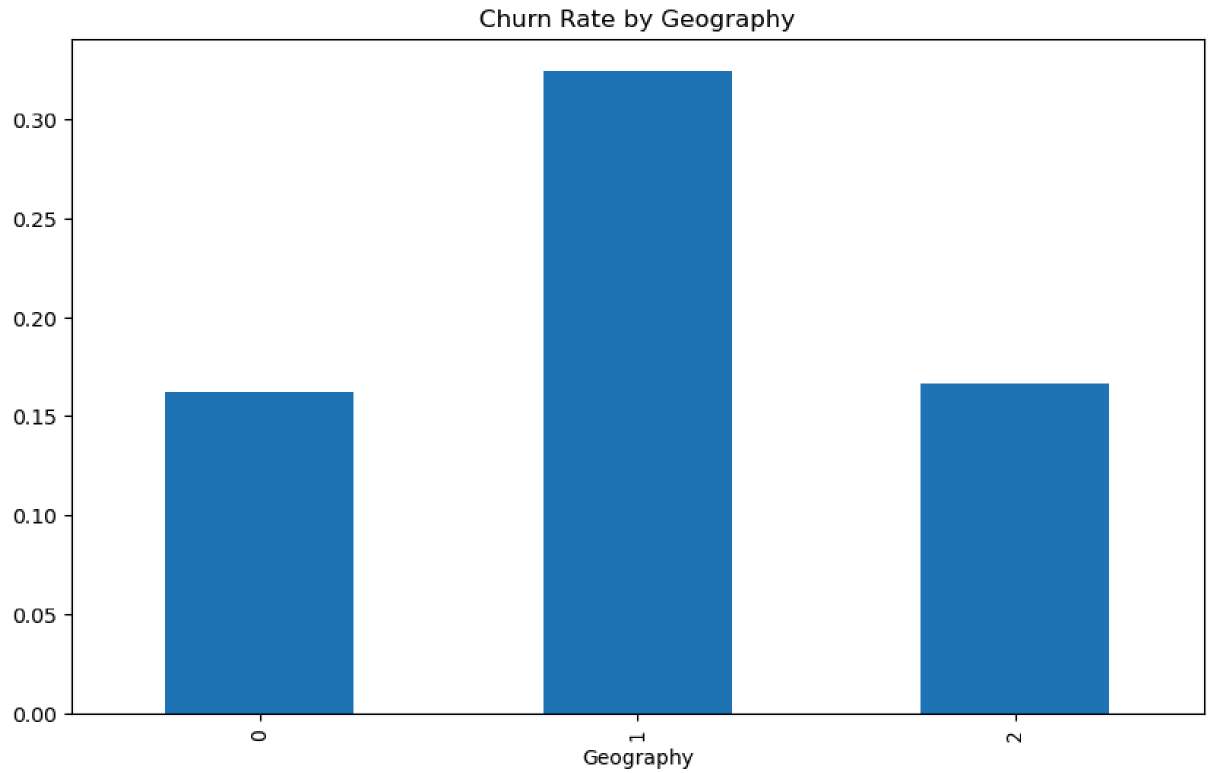
In [48]:
```
#Age Analysis

# Plot the distribution of 'Age' for churned and non-churned customers
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', hue='Exited', kde=True, multiple='stack')
plt.title('Age Distribution by Churn Status')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



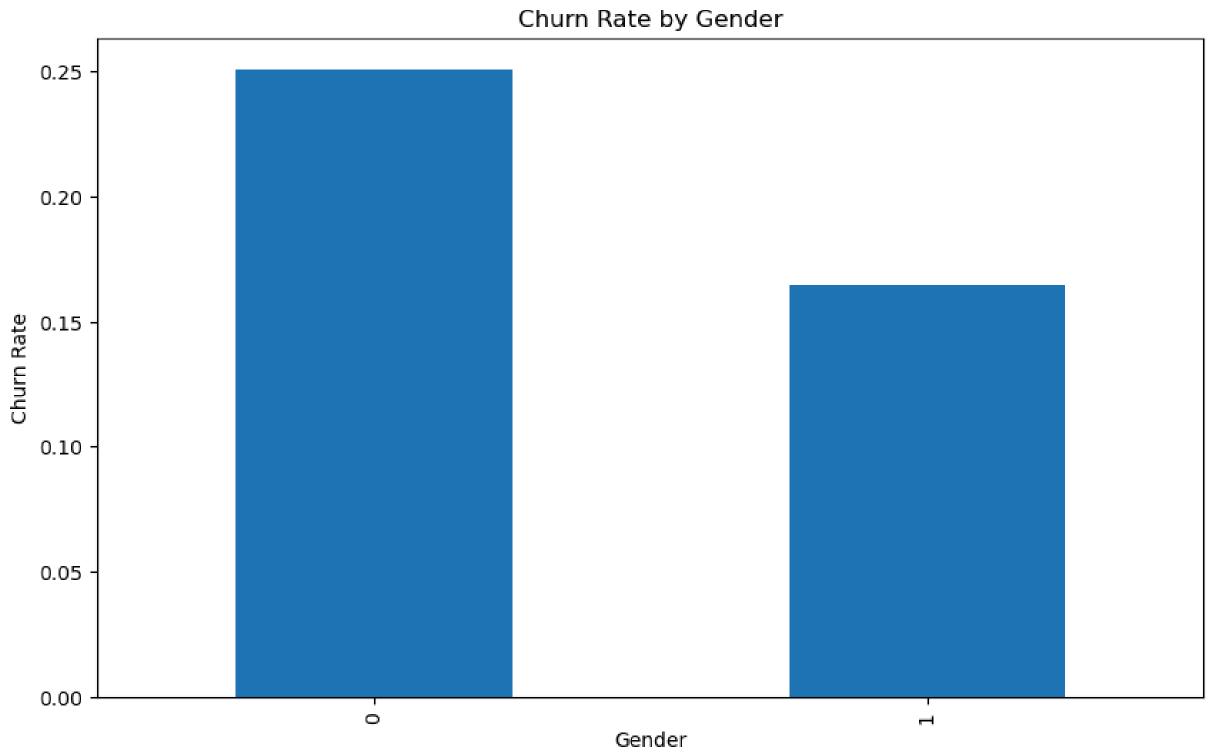
In [49]:
```
#Geography Analysis


# Plot the churn rate by Geography
geo_churn_rate = df.groupby('Geography')['Exited'].mean()
plt.figure(figsize=(10, 6))
geo_churn_rate.plot(kind='bar')
plt.title('Churn Rate by Geography')
plt.xlabel('Geography')
```

Out[49]:
```
Text(0.5, 0, 'Geography')
```

Loading [MathJax]/extensions/Safe.js



In [50]:
```
#Gender Analysis

# Plot the churn rate by Gender
gender_churn_rate = df.groupby('Gender')['Exited'].mean()
plt.figure(figsize=(10, 6))
gender_churn_rate.plot(kind='bar')
plt.title('Churn Rate by Gender')
plt.xlabel('Gender')
plt.ylabel('Churn Rate')
plt.show()
```



In [ ]:

In [ ]:

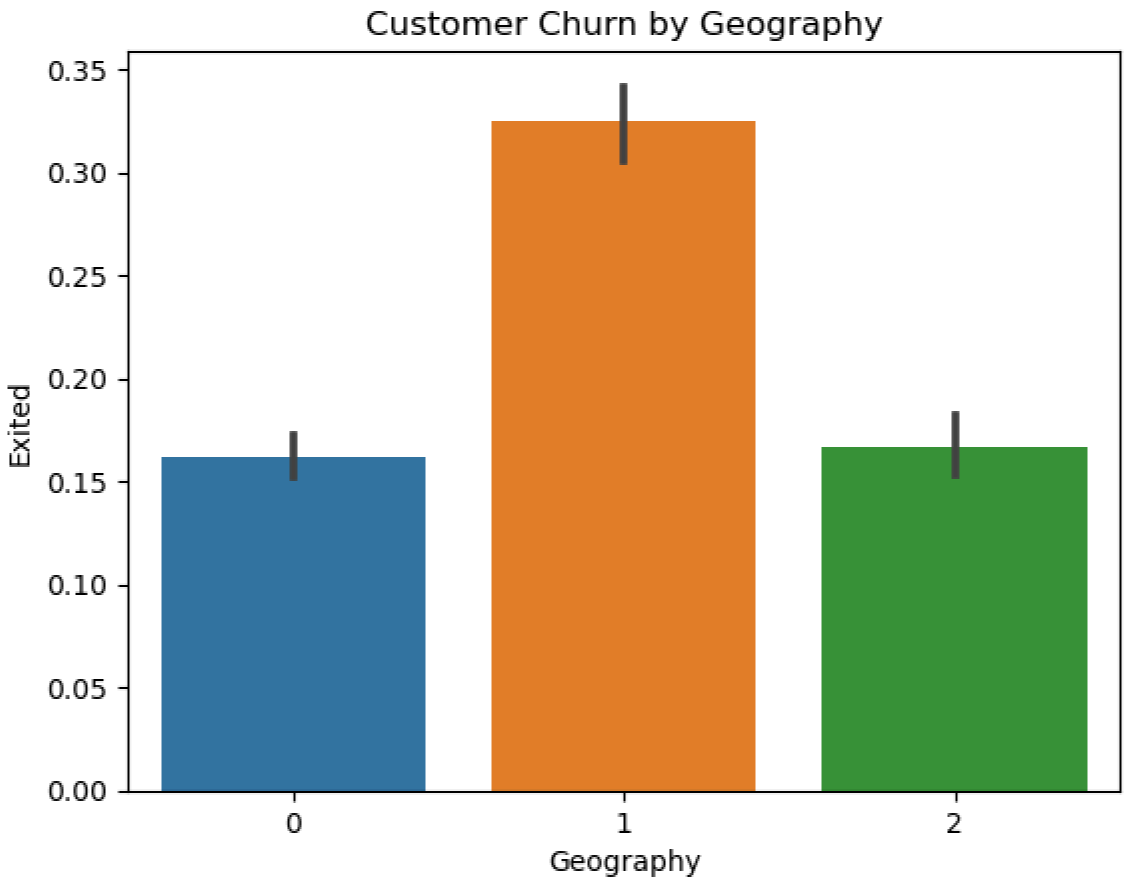
In [52]:
```
#Customer Profile Analysis: Segment customers based on key demographics (Age

#By groupby method
Geography_Churn=df.groupby('Geography')['Exited'].count().reset_index()
Geography_Churn
```

Out[52]:

	Geography	Exited
0	0	5014
1	1	2509
2	2	2477

In [79]:
```
sns.barplot(x='Geography',y='Exited',data=df)
plt.title('Customer Churn by Geography')
plt.show()
```

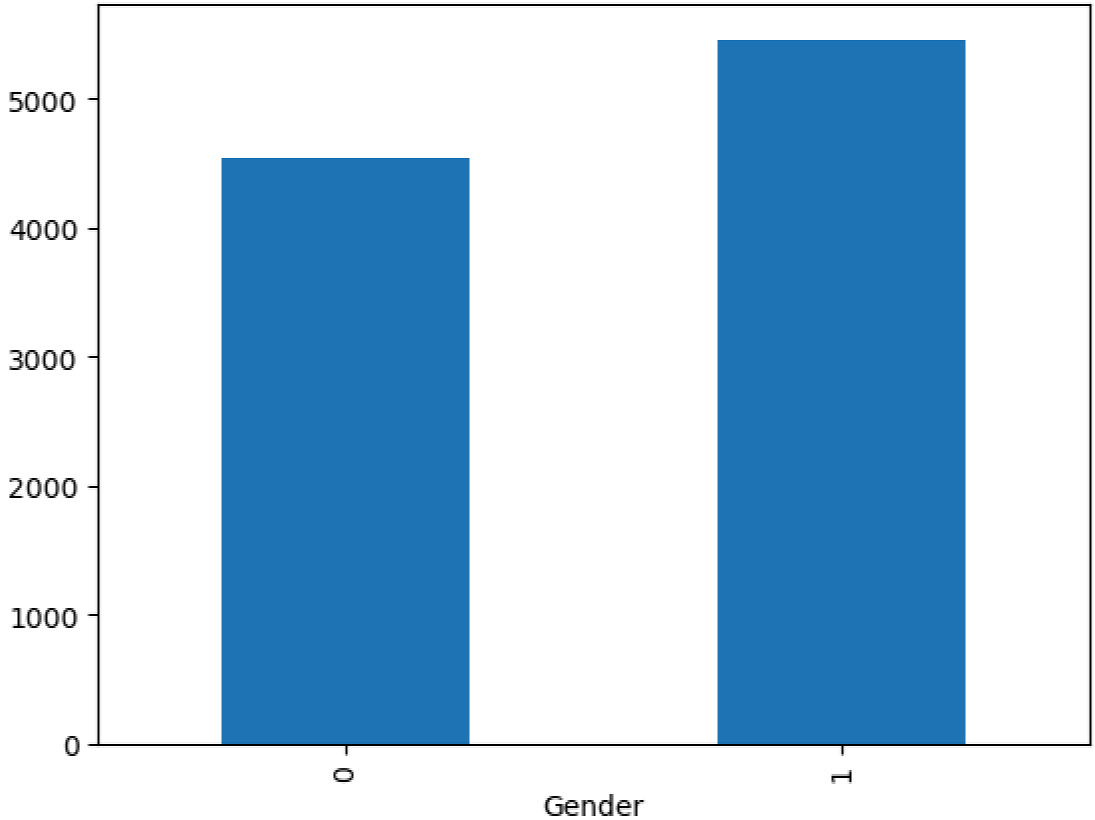
Loading [MathJax]/extensions/Safe.js



In [ ]:
```
Majority of the people from France. However, the proportion of churned custo
in the areas where it has fewer clients.
```

In [72]:
```
# Gender Differences in Churn: Analyze churn rates between different genders

churn_rate_by_gender = df.groupby(['Gender'])['Exited'].count()
print(churn_rate_by_gender)
churn_rate_by_gender.plot(kind ='bar')
plt.show()
```

```
Gender
0    4543
1    5457
Name: Exited, dtype: int64
```



In [73]:

```
# lets do a chi square test to identyfy if gender plays a significant role i

contingency_table  = pd.crosstab(index = df['Gender'], columns =df['Exited']
print(contingency_table)

res = chi2_contingency(contingency_table)
print(res.pvalue)

if res.pvalue < 0.05:
  print('Gender has significant role in churn')
else:
  print('Gender has NO significant role in churn')
```

```
Exited     0     1
Gender
0       3404  1139
1       4558   899
2.9253677618642e-26
Gender has significant role in churn
```

In [81]:

```
#By group by method
Age_Churn=df.groupby('Age')['Exited'].count().reset_index()
Age_Churn
```

Loading [MathJax]/extensions/Safe.js

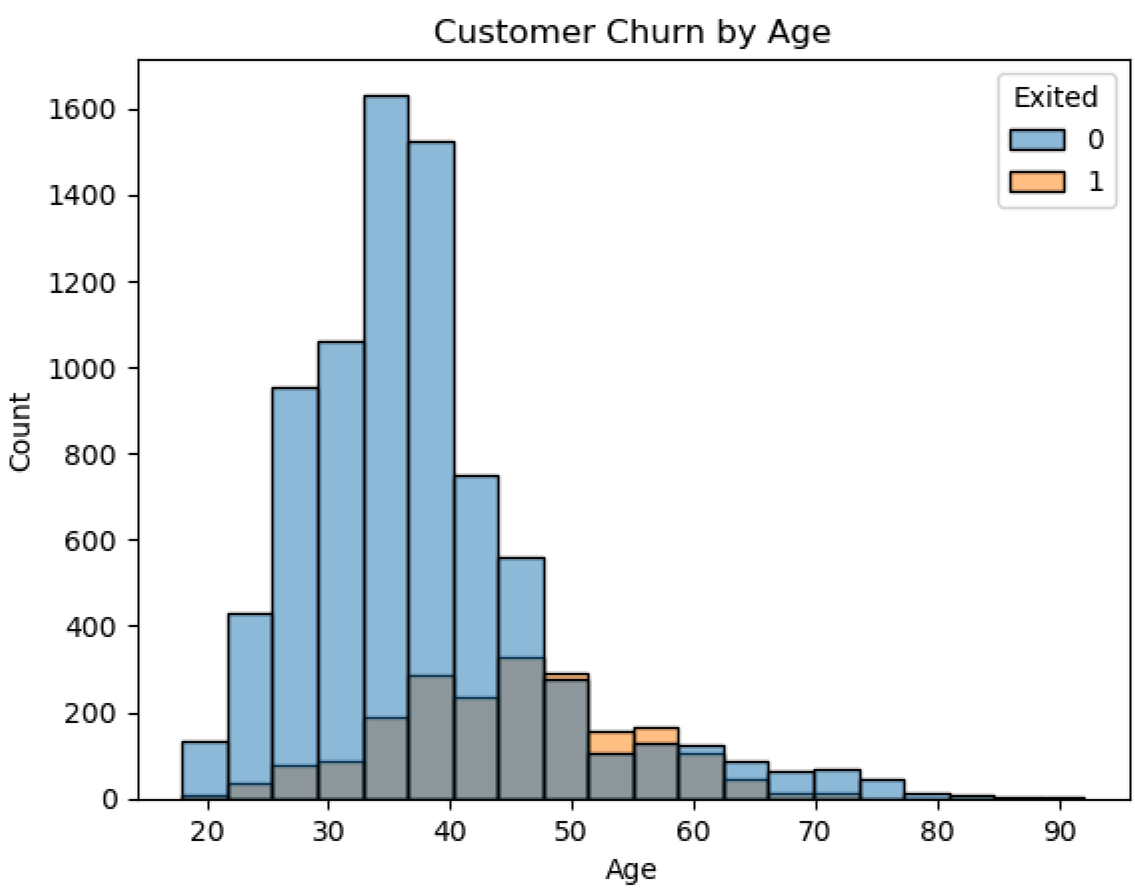
Out[81]:

	Age	Exited
0	18	22
1	19	27
2	20	40
3	21	53
4	22	84
...	...	...
65	83	1
66	84	2
67	85	1
68	88	1
69	92	2

70 rows × 2 columns

In [83]:

```
#Using histplot
sns.histplot(hue='Exited',x='Age',data=df,bins=20)
plt.title('Customer Churn by Age')
plt.show()
```



In [ ]:
```
The older customers are churning at more than the younger ones incuding to a
The bank may need to review their target market or review the strategy for r
```

In [85]:
```
#By group by Method
Gender_Churn=df.groupby('Gender')['Exited'].count().reset_index()
Gender_Churn
```

Out[85]:

	Gender	Exited
0	0	4543
1	1	5457

In [ ]:
```
The proportion of female customers churning is also greater than that of mal
```

In [87]:
```
#3. Comparative Analysis

#Churn by Geography: Compare churn rates across different geographical locat

#By Hypothesis testing

cross_table = pd.crosstab(df['Geography'],df['Exited'])
print(cross_table)
```

```
Exited        0    1
Geography
0          4203  811
1          1695  814
2          2064  413
```

In [ ]:
```
By comparing the churn rates with geographical locations, Germany has higher
```

In [88]:
```
from scipy.stats import chi2_contingency,chi2

#h0: There is no association between geography and churn rates.
#h1: There is association between geography and churn rates.

chi2_stat,p_val,dof,expected = chi2_contingency(cross_table)
print(p_val)

alpha=0.05
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
5.245736109572763e-66
Reject H0
```

In [ ]:
```
There is association between geography and churn rates.
```

In [90]:
```
#Gender Differences in Churn: Analyze churn rates between different genders
#By Hypothesis Testing

Loading [MathJax]/extensions/Safe.js is no association between gender and churn.
```

```
#h1:There is association between gender and churn.

churn_table= pd.crosstab(df['Gender'],df['Exited'])
print('observed values:')
churn_table
```

observed values:

Out[90]:

Exited	0	1
Gender		
0	3404	1139
1	4558	899

In [ ]:

```
On comparing the churn rate with genders Female has more churn rating than M
```

In [92]:

```
#Chisquare test
stats,p_val,dof,expected=chi2_contingency(churn_table)
print("t_statistics:",stats)
print("p_value:",p_val)

alpha=0.05
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
t_statistics: 112.39655374778587
p_value: 2.9253677618642e-26
Reject H0
```

In [ ]:

```
There is association between gender and churn.
```

In [93]:

```
#4. Behavioral Analysis

#Product and Services Usage: Examine how the number of products (NumOfProduc

#H0=NumOfProducts has no significant effect on the likelihood to churn.
#H1=NumOfProducts has significant effect on the likelihood to churn.

from scipy.stats import chi2_contingency,chi2

data_table = pd.crosstab(df['NumOfProducts'], df['Exited'])
print("Observed values:")
data_table
```

Observed values:

Out[93]:

Exited	0	1
NumOfProducts		
1	3675	1409
2	4241	349
3	46	220
4	0	60

In [94]:

```
alpha=0.05

stats, p_val,dof,expected=chi2_contingency(data_table)
print("test statistic:",stats)
print("p_val:",p_val)
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
test statistic: 1501.5048306588592
p_val: 0.0
Reject H0
```

In [ ]:

```
Reject Null Hypothesis: It mean there is significant effect of NumOfProducts
```

In [95]:

```
df.groupby(by='NumOfProducts')['Exited'].count().reset_index()
```

Out[95]:

	NumOfProducts	Exited
0	1	5084
1	2	4590
2	3	266
3	4	60

In [ ]:

```
Product has no significant effect on the likelihood to churn.
```

In [97]:

```
#Activity Level Analysis: Investigate the relationship between being an IsAc

#By Hypothesis testing

#h0:there is no relationship between IsActiveMember and customer Churn
#h1:there is relationship between IsActiveMember ans customer Churn

contingency_table=pd.crosstab(df['IsActiveMember'],df['Exited'])
print(contingency_table)
```

```
Exited            0     1
IsActiveMember
0              3546  1303
1              4416   735
```

In [98]:
```
stats,p_val,dof,expected=chi2_contingency(contingency_table)
print("t_statistics:",stats)
print("p_value:",p_val)

alpha=0.05
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

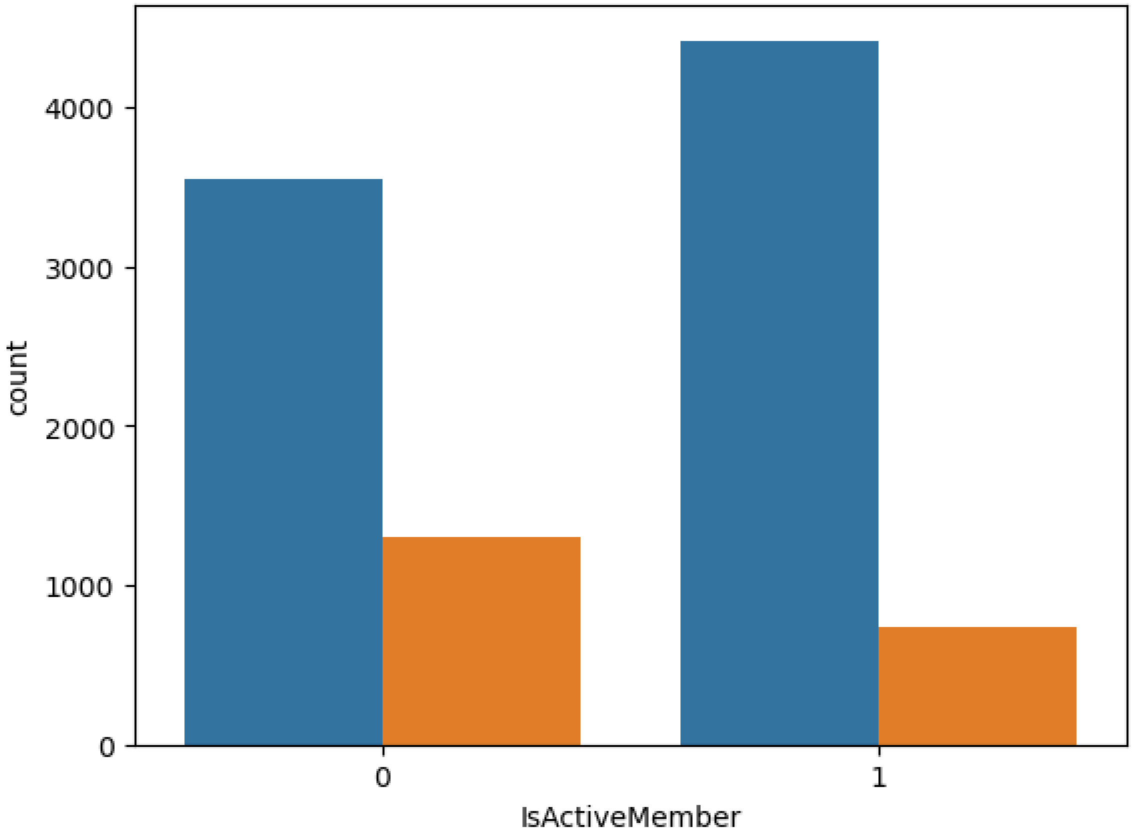
```
t_statistics: 243.6948024819593
p_value: 6.153167438113408e-55
Reject H0
```

In [ ]:
```
A significant p-value indicates that the churn rate is dependent on the acti
```

In [99]:
```
#Using Countplot
sns.countplot(x='IsActiveMember',hue='Exited',data=df)
plt.title('Customer Churn by IsActiveMember')
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[99], line 2
      1 #Using Countplot
----> 2 sns.countplot(x='IsActiveMember',hue='Exited',data=df)
      3 plt.title('Customer Churn by IsActiveMember')
      4 plt.show()

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:2955, in countplot
(data, x, y, hue, order, hue_order, orient, color, palette, saturation, widt
h, dodge, ax, **kwargs)
   2952 if ax is None:
   2953     ax = plt.gca()
-> 2955 plotter.plot(ax, kwargs)
   2956 return ax

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:1587, in _BarPlott
er.plot(self, ax, bar_kws)
   1585 """Make the plot."""
   1586 self.draw_bars(ax, bar_kws)
-> 1587 self.annotate_axes(ax)
   1588 if self.orient == "h":
   1589     ax.invert_yaxis()

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:767, in _Categoric
alPlotter.annotate_axes(self, ax)
    764     ax.set_ylim(-.5, len(self.plot_data) - .5, auto=None)
    766 if self.hue_names is not None:
--> 767     ax.legend(loc="best", title=self.hue_title)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:322, in Axes.leg
end(self, *args, **kwargs)
    204 @_docstring.dedent_interpd
    205 def legend(self, *args, **kwargs):
    206     """
    207     Place a legend on the Axes.
    208
   (...)
    320     .. plot:: gallery/text_labels_and_annotations/legend.py
    321     """
--> 322     handles, labels, kwargs = mlegend._parse_legend_args([self], *ar
gs, **kwargs)
    323     self.legend_ = mlegend.Legend(self, handles, labels, **kwargs)
    324     self.legend_._remove_method = self._remove_legend

File ~\anaconda3\Lib\site-packages\matplotlib\legend.py:1361, in _parse_lege
nd_args(axs, handles, labels, *args, **kwargs)
   1357     handles = [handle for handle, label
   1358                in zip(_get_legend_handles(axs, handlers), labels)]
   1360 elif len(args) == 0:  # 0 args: automatically detect labels and hand
les.
-> 1361     handles, labels = _get_legend_handles_labels(axs, handlers)
   1362     if not handles:
   1363         log.warning(
   1364             "No artists with labels found to put in legend.  Note th
```

```
   1365             "artists whose label start with an underscore are ignore
d "
   1366             "when legend() is called with no argument.")

File ~\anaconda3\Lib\site-packages\matplotlib\legend.py:1291, in _get_legend
_handles_labels(axs, legend_handler_map)
   1289 for handle in _get_legend_handles(axs, legend_handler_map):
   1290     label = handle.get_label()
-> 1291     if label and not label.startswith('_'):
   1292         handles.append(handle)
   1293         labels.append(label)

AttributeError: 'numpy.int64' object has no attribute 'startswith'
```



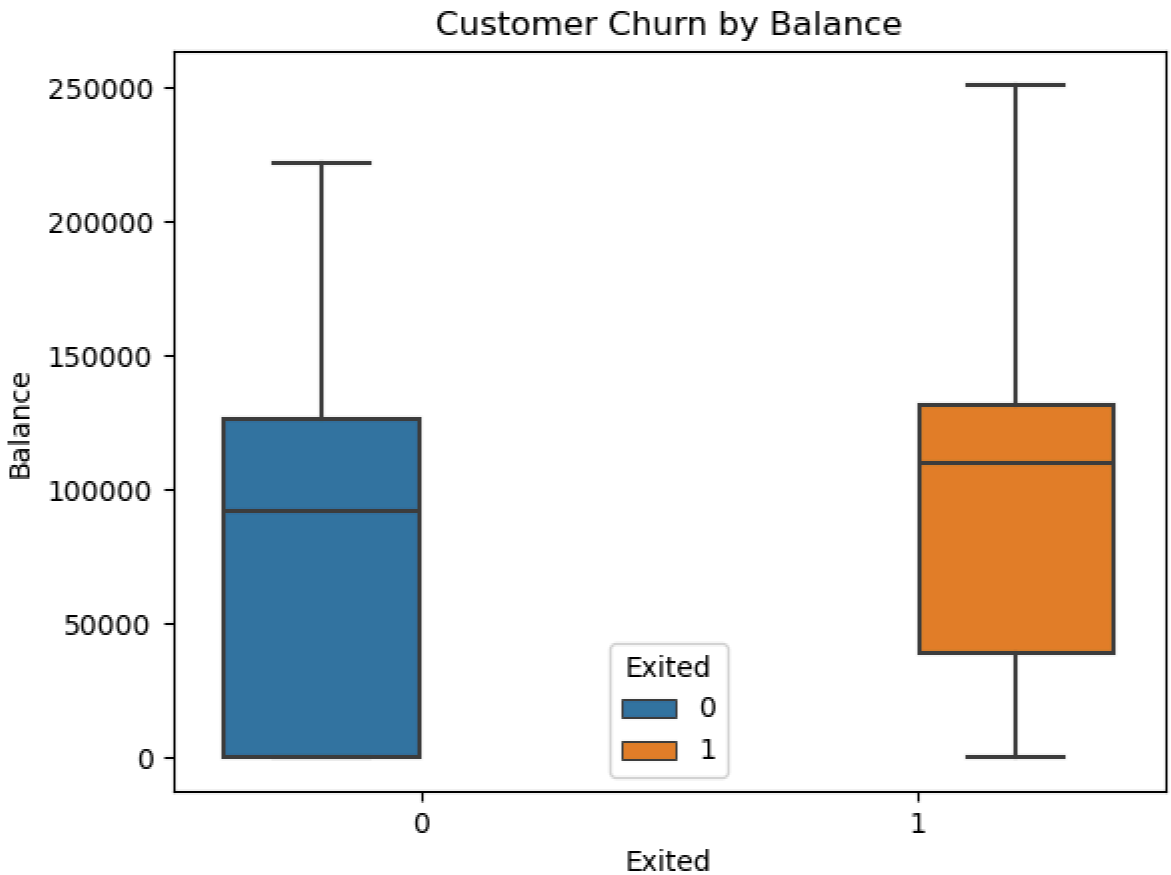
In [ ]:

```
Showed relation between IsActiveMember and churned customers in both ways Hy
Unsurprisingly the inactive members have a greater churn.
The overall proportion of inactive mebers is quite high suggesting that the
customers as this will definately have a positive impact on the customer chu
```

In [100…

```
#5. Financial Analysis

#Balance vs. Churn: Analyze how customer balance levels correlate with churn

#using Boxplot
sns.boxplot(x='Exited',y='Balance',hue='Exited',data=df)
plt.title('Customer Churn by Balance')
plt.show()
```

Loading [MathJax]/extensions/Safe.js



In [102…
```
#By Hypothesis testing- T test
data_0= df[df['Exited']==0]
data_1 = df[df['Exited']==1]
```

In [ ]:
```
#By Hypothesis testing- T test
data_0= df[df['Churned']==0]
data_1 = df[df['Churned']==1]
```

In [ ]:

In [ ]:

In [ ]:

In [104…
```
#H0:Balance has no significant effect on the likelihood to churn.
#H1:Balance has significant effect on the likelihood to churn.

t_statistic, p_value = ttest_ind(data_0['Balance'], data_1['Balance'],alterr
print("t_statistics:",t_statistic)
print("p_value:",p_value)

alpha=0.05
if p_value<= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
t_statistics: -11.940747722508185
p_value: 1.2092076077156017e-32
Reject H0
```

In [ ]:
```
As p_val< alpha, Balance has sigincant effecr on the likelihood to churn.Wor
the bank is losing customers with significant bank balances which is likely
```

In [ ]:
```
Interestingly, majority of the customers that churned are those with credit
Given that majority of the customers have credit cards could prove this to b
```

In [108…
```
#Chi2 test

#h0:HasCrCard has no significant effect on the likelihood to churn.
#h1:HasCrCard has significant effect on the likelihood to churn.

cross_table=pd.crosstab(df['HasCrCard'],df['Exited'])
print('Observed values:')
print(cross_table)
stats,p_val,dof,expected=chi2_contingency(cross_table)
print("t_statistics:",stats)
print("p_value:",p_val)

alpha=0.05
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
Observed values:
Exited        0     1
HasCrCard
0          2332   613
1          5630  1425
t_statistics: 0.4494039375253385
p_value: 0.5026181509009862
Accept H0
```

In [ ]:
```
There is no significant difference on owing a Creditcard for churning the Ba
```

In [ ]:
```
```

There are significantly more customers who did not complain than those who did.Among those who did complain, a higher proportion of them churned compared to those who did not complain.

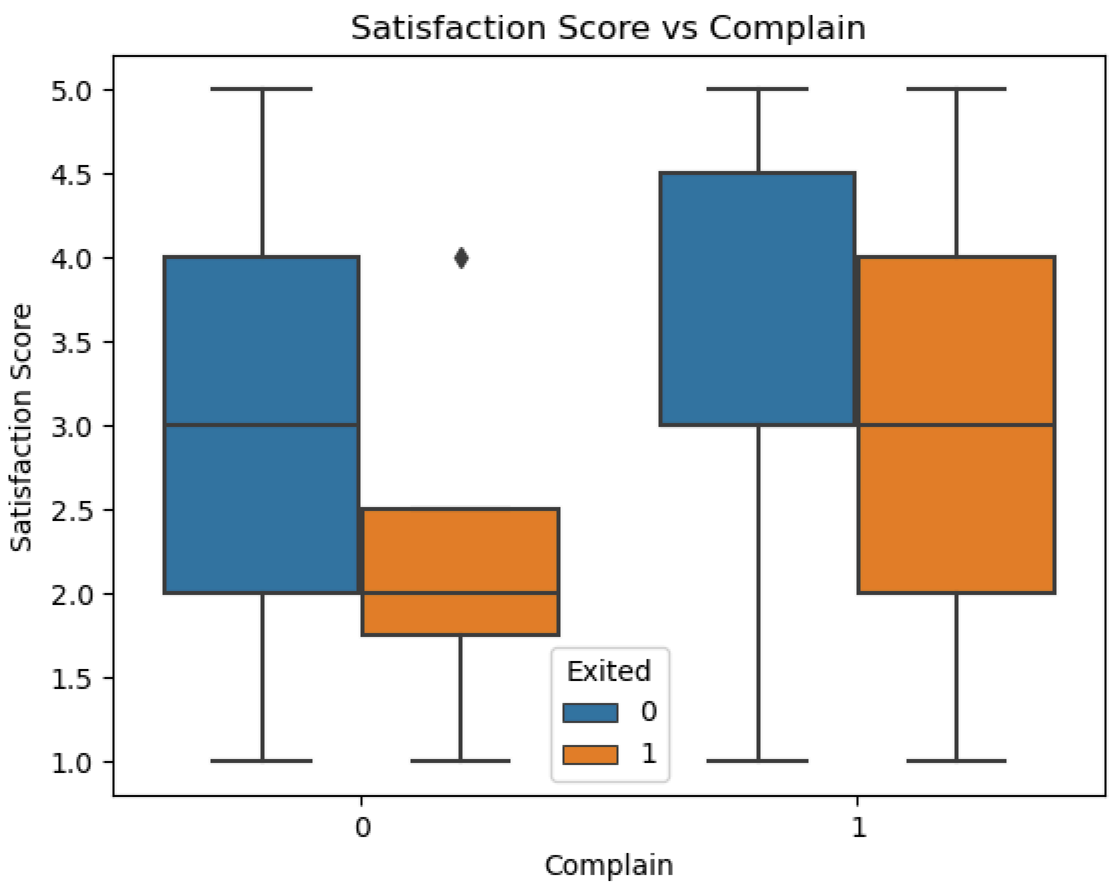
In [111…
```
#T test


#H0: Complain has no significant effect on the likelihood to churn.
#H1: Complain has significant effect on the likelihood to churn.

t_statistic, p_value = ttest_ind(data_0['Complain'], data_1['Complain'],alte
print("t_statistics:",t_statistic)
print("p_value:",p_value)
```

```
alpha=0.05
if p_value<= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
t_statistics: -1073.7975930429423
p_value: 0.0
Reject H0
```

In [ ]:
```
There is a significant difference on churn rate by Complains.Customers who c
```

In [113…
```
#Satisfaction and Churn: Explore how the Satisfaction Score relates to churn

#Using Boxplot
sns.boxplot(y='Satisfaction Score',x='Complain',hue='Exited',data=df)
plt.title('Satisfaction Score vs Complain')
plt.show()
```



In [ ]:
```
```

In [115…
```
#Two way Anova test


#import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

Loading [MathJax]/extensions/Safe.js

```python
import seaborn as sns
import matplotlib.pyplot as plt

model = ols('Q("Satisfaction Score") ~ C(Exited) * C(Complain)', data=df).fi
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)

# Interpret the results
if anova_table["PR(>F)"].iloc[0] < 0.05:
    print("Main effect of Churned is significant.")
else:
    print("Main effect of Churned is not significant.")

if anova_table["PR(>F)"].iloc[1] < 0.05:
    print("Main effect of Complain is significant.")
else:
    print("Main effect of Complain is not significant.")

if anova_table["PR(>F)"].iloc[2] < 0.05:
    print("Interaction effect between Churned and Complain is significant.")
else:
    print("Interaction effect between Churned and Complain is not significar
```

```
                              sum_sq      df         F    PR(>F)
C(Exited)                   2.636157     1.0  1.333528  0.248206
C(Complain)                 2.415155     1.0  1.221732  0.269048
C(Exited):C(Complain)       0.621009     1.0  0.314144  0.575161
Residual                19760.383245  9996.0       NaN       NaN
Main effect of Churned is not significant.
Main effect of Complain is not significant.
Interaction effect between Churned and Complain is not significant.
```

In [ ]:

In [ ]:

```python
#7. Card Usage Analysis

#Impact of Card Type on Churn: Examine if different Card Types have differer

sns.countplot(x='Card Type',hue='Churned',color='lightblue',data=df)
plt.title('Card Usage Analysis')
plt.xlabel('Card Type')
plt.ylabel('Count')
plt.show()
```

In [117…

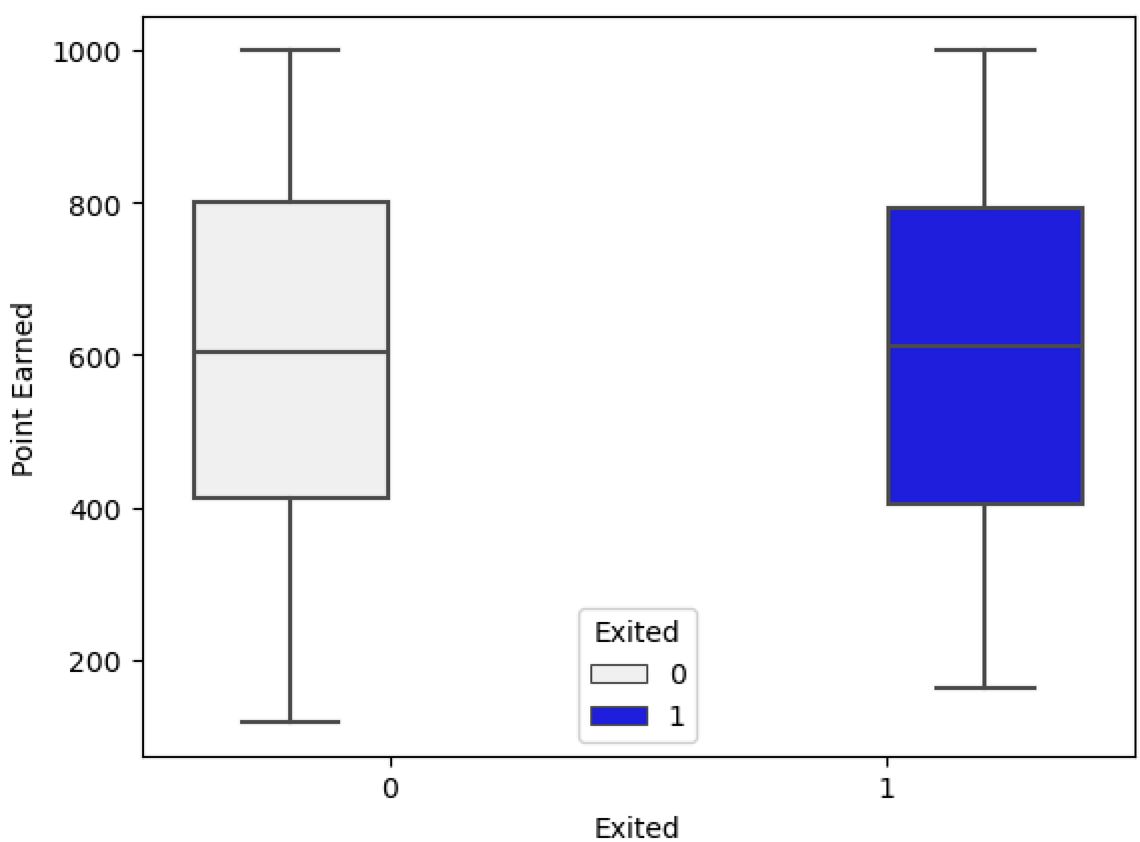
```python
#Impact of Card Type on Churn: Examine if different Card Types have differer

#ho=Card type has no diff churn rates
#h1=Card type has diff churn rates

#Chi Square test

from scipy.stats import chi2_contingency

cross_table=pd.crosstab(df['Card Type'],df['Exited'])
print('Observed values:')
print(cross_table)
```

```
t_statistic,p_val,dof,expected=chi2_contingency(cross_table)
print("t_statistics:",t_statistic)
print("p_value:",p_val)

alpha=0.05
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
Observed values:
Exited       0    1
Card Type
0          1961  546
1          2020  482
2          1987  508
3          1994  502
t_statistics: 5.053223027060927
p_value: 0.16794112067810177
Accept H0
```

In [ ]:
```
There is no such difference based on card type on the customer churn.
```

In [127…
```
#Loyalty Points Analysis: Investigate whether Points Earned from credit card

sns.boxplot(y='Point Earned',x='Exited',hue='Exited',color='blue',data=df)
plt.ylabel('Point Earned')
plt.show()
```

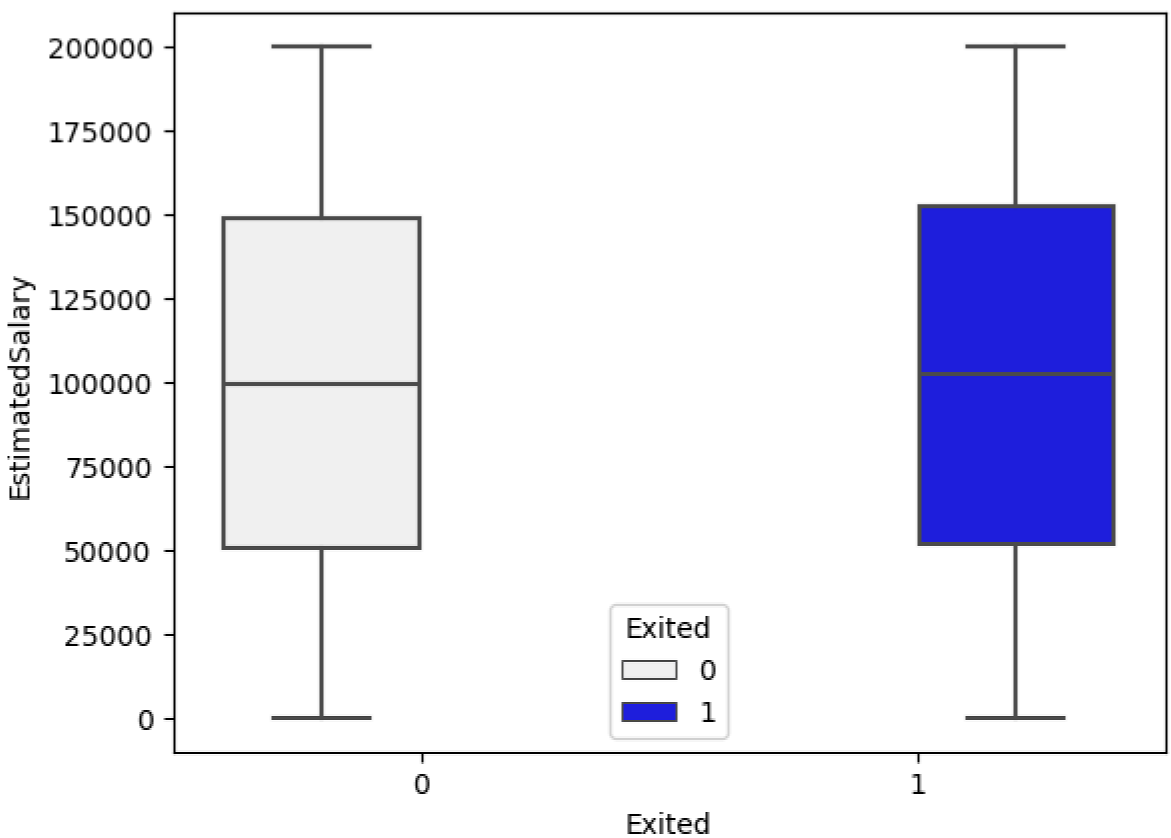


In [ ]:
```
Whatever the points earned by Customers, that is not related to the customer
```

In [122…
```
#H0:Points Earned has no significant effect on the likelihood to churn.
#H1:Points Earned has significant effect on the likelihood to churn.

stats,p_val=ttest_ind(data_0['Point Earned'],data_1['Point Earned'],alternat
print("t_statistics:",stats)
print("p_value:",p_val)

alpha=0.05
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
t_statistics: 0.4627759848070133
p_value: 0.6435350184288993
Accept H0
```

In [126…
```
#8. Salary Analysis

#Salary and Churn: Analyze the relationship between EstimatedSalary and cust

#Using boxplot
sns.boxplot(y='EstimatedSalary',x='Exited',hue='Exited',color='blue',data=df
plt.show()
```



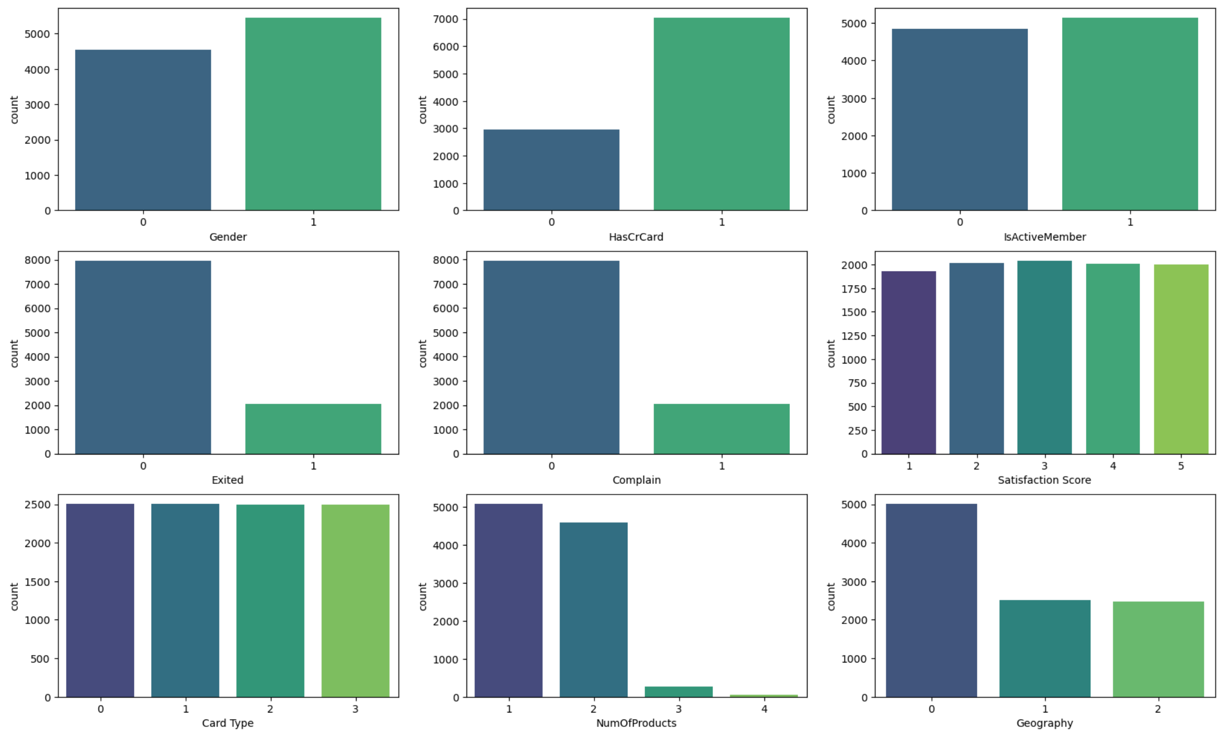
In [ ]:
```
The salary has no significant effect on the likelihood to churn.
```

In [131…
```python
#T test

#H0:EstimatedSalary has no significant effect on the likelihood to churn.
#H1:EstimatedSalary has significant effect on the likelihood to churn.

from scipy.stats import ttest_ind

data_0=df[df['Exited']==0]
data_1=df[df['Exited']==1]

stats,p_val=ttest_ind(data_0['EstimatedSalary'],data_1['EstimatedSalary'],al
print("t_statistics:",stats)
print("p_value:",p_val)

alpha=0.05
if p_val <= alpha:
    print("Reject H0")
else:
    print("Accept H0")
```

```
t_statistics: -1.2489445044833742
p_value: 0.2117146135149097
Accept H0
```

In [ ]:

In [138…
```python
fig, axs = plt.subplots(nrows = 3, ncols = 3, figsize = (20,12))
sns.countplot(data = df, x = 'Gender', ax = axs[0,0], palette = 'viridis')
sns.countplot(data = df, x = 'HasCrCard', ax = axs[0,1], palette = 'viridis'
sns.countplot(data = df, x = 'IsActiveMember', ax = axs[0,2], palette = 'vir
sns.countplot(data = df, x = 'Exited', ax = axs[1,0], palette = 'viridis')
sns.countplot(data = df, x = 'Complain', ax = axs[1,1], palette = 'viridis')
sns.countplot(data = df, x = 'Satisfaction Score', ax = axs[1,2], palette =
sns.countplot(data = df, x = 'Card Type', ax = axs[2,0], palette = 'viridis'
sns.countplot(data = df, x = 'NumOfProducts', ax = axs[2,1], palette = 'viri
sns.countplot(data = df, x = 'Geography', ax = axs[2,2], palette = 'viridis'
plt.show()
```



In [187…

```
result = df.groupby('Exited')['Balance'].agg(['mean', 'median'])
print(result)
```

```
                                                           mean     median
Exited
<bound method NDFrame.astype of       NumOfProd...  76485.889288  97198.54
```

In [ ]:

```
∇ Age Vs Customer Churn
Null Hypothesis : There is no significant difference between the mean age of
Alternative Hypothesis : There is significant difference between the mean ag
```

In [212…

```
a_exited = df[df['Exited'] == 1]['Age']
a_stayed = df[df['Exited'] == 0]['Age']
alpha = 0.05
stats, pval = ttest_ind(a_stayed, a_exited, equal_var = False)
if pval < alpha :
 print('Reject Null Hypothesis')
else:
 print('Failed to Reject Null Hypothesis')
```

```
Failed to Reject Null Hypothesis
```

In [ ]:

```
There is significant difference between the mean age of the customer who exi
```

In [ ]:

In [ ]:

```
Balance Vs Customer Churn
Null Hypothesis : There is no significant difference between the mean balanc
Alternative Hypothesis : There is significant difference between the mean ba
```

In [214…

```
b_exited = df[df['Exited'] == 1]['Balance']
b_stayed = df[df['Exited'] == 0]['Balance']
.05
```

```python
stats, pval = ttest_ind(b_stayed, b_exited, equal_var = False)
if pval < alpha :
 print('Reject Null Hypothesis')
else:
 print('Failed to Reject Null Hypothesis')
```

```
Failed to Reject Null Hypothesis
```

In [ ]:
```python
There is significant difference between the mean balance of the customer whc
```

In [ ]:
```python

```

Insights:

In [ ]:
```python

```

Expand Marketing Efforts in Germany and Spain: Since 50% of customers are from France, focus marketing campaigns on Germany and Spain to boost customer acquisition in these regions.

Develop Targeted Offers for Female Customers: Introduce specific products or offers aimed at attracting more female customers to balance the customer demographics.

Enhance After-Sales Service: Address the fact that almost 99% of customers who filed complaints have left the bank by significantly improving the after-sales service experience.

Create Retention Strategies for Multi-Product Holders: Implement targeted retention strategies for customers with three or more products, as they have a higher churn rate.

Engage Zero Balance Account Holders: Investigate why approximately 3,000 accounts have zero balance and develop offers or incentives to engage these customers and encourage account usage.

Financial Counseling for At-Risk Customers: Analyze factors influencing customer exit versus retention and offer financial counseling to customers in vulnerable salary brackets to reduce churn

Recommendation:-

In [ ]:
```python
Target customers between 40-50 age group with personalized retention strateg

Analyze churn rates by region and implement targeted strategies for high-ris

improving the experience of female customers to reduce their slight
```

Loading [MathJax]/extensions/Safe.js

```
Encourage customers to use more products, especially those with 1-2 products

Engage customers and promote active use of services to reduce churn. Emphasi

Investigate and address the reasons behind customers having zero balance or

Enhance complaint resolution mechanisms to address customer concerns and rec

Regularly monitor satisfaction scores and address any areas where customers
reduce churn.

Analyze the impact of different card types on churn and consider offering ir
loyalty.

Explore alternative loyalty programs or incentives to increase customer enga
```

In [ ]: